

MAYOR OF LONDON

**THE LONDON CURRICULUM
COMPUTING KEY STAGE 3**

THE CONNECTED CITY



COMPUTING AT SCHOOL
EDUCATE · ENGAGE · ENCOURAGE
Part of BCS, The Chartered Institute for IT

UCL
Institute of Education

THE LONDON CURRICULUM

PLACING LONDON AT THE HEART OF LEARNING

The capital is the home of innovations, events, institutions and great works that have extended the scope of every subject on the school curriculum. London lends itself to learning unlike anywhere else in the world. The London Curriculum aims to bring the national curriculum to life inspired by the city, its people, places and heritage.

To find out about the full range of free resources and events available to London secondary schools at key stage 3 please go to: www.london.gov.uk/london-curriculum.

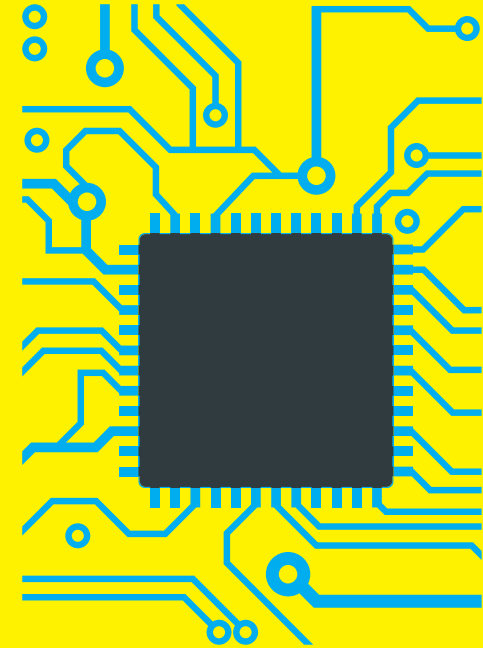
STEM in the London Curriculum

London provides numerous historical and contemporary cutting edge examples of scientists, engineers and mathematicians who have worked in their fields to create innovative solutions to problems throughout the world. Population growth, trade, communication, transport, health, food, water supply and many other aspects of life in London have driven technology-based innovations. London Curriculum science, maths, design & technology teaching resources aim to support teachers in helping their students to:

DISCOVER the application of their subject knowledge to the life of the city.

EXPLORE their neighbourhood and key sites around London, learning outside the classroom to see and understand how STEM subjects have shaped many aspects of the city.

CONNECT their learning inside and outside the classroom, analysing situations and using their subject knowledge to create and present solutions.



CONTENTS

OVERVIEW	2	Lesson 2: Traffic sensing technology	28	EXPLORE	98
CONDUCTING A RISK ASSESSMENT	3	Activities	31	Lesson 5: Exploring transport systems	99
<hr/>		Student scratch activities	35	Activities	100
DISCOVER	8	Resource 2.1: Pedestrian crossing unplugged (with interruption)	44	<hr/>	
Lesson 1: Controlling London's transport	9	Resource 2.2: Green man / red man crossing cards	45	CONNECT	103
Activities	13	Resource 2.3: Car and traffic light cards	46	Lesson 6: Computing London solutions	104
Student Scratch activities	16	Resource 2.4: Algorithm cards	50	Activities	106
Resource 1.1: Every journey matters	23	<hr/>		Resource C1: Reporting framework example – congestion charge	108
Resource 1.2: The first traffic light	25	Lesson 3: Where's the bus?	51	<hr/>	
Resource 1.3: Teacher instructions for Algorithm Card	26	Student Python activities	53	Links to other London Curriculum subjects	110
Resource 1.4: Student Algorithm Cards	27	<hr/>		Credits	111
<hr/>		Lesson 4: You're the developer	74		
		Student Python activities	76		
		<hr/>			

UNIT OVERVIEW

UNIT AIMS AND OBJECTIVES

Over 26 million trips are made on an average day in London, and that number is growing year after year. Improving the efficiency of London's transport leads not only to savings in time and costs but also other important benefits such as reduced energy consumption and air pollution.

Students will look into the computing behind the road traffic signals and see how their learning is applied in the real life of the city, which might include visits and interactions with Transport for London engineers.

As well as simple mechanical alternation of traffic at junctions, computer sensing and data collection can be used to control traffic flow on a larger scale, manage congestion, change driver behaviour and share the roads more evenly between different users such as cars and cyclists.

The overarching theme is of computational thinking as an approach to represent and solve the problems involved in managing and improving traffic flow in London. Students will see that this is at the heart of decision making and engineering computing solutions in the real world.



OXFORD CIRCUS DIAGONAL CROSSING
© Transport for London



CONDUCTING A RISK ASSESSMENT

For learning outside the classroom

It is the responsibility of each institution, delegated to the class teacher, to make risk assessments for a given class and a given location. Guidance can be found through the membership organisation CLEAPSS for all school science.

www.cleapss.org.uk

More general guidance on risk assessment for school trips can be found here:

www.atl.org.uk/health-and-safety/off-site-trips/risk-assessment-school-trips.asp

For practical work

A general guide for health and safety guidance and risk assessment of practical work can be found here:

www.nuffieldfoundation.org/standard-health-safety-guidance

If any additional specific guidance is necessary for particular practicals, this will be found within the instructions for each practical.

KEY STAGE 3 NATIONAL CURRICULUM

This unit addresses the following general aims, that students can:

- ◆ understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation
- ◆ analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
- ◆ evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems and the following subject content of the curriculum:
- ◆ design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- ◆ use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures (for example, lists, tables or arrays)
- ◆ understand simple Boolean logic (for example, AND, OR and NOT) and some of its uses in circuits and programming
- ◆ understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems

- ◆ undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users

Python and Scratch programmes to support the Connected City unit

Example programs for the problems are provided in Scratch for lessons 1 and 2 and in Python for lessons 3 and 4. Students are expected to be able to open, edit and run programs in Scratch and Python, and also in Python be able to create a new program. They will be both modifying the example programs and writing new programs themselves. Example programs can be accessed from the London Curriculum teacher's hub, to which all registered teachers have access.

For lesson 3 we have supplied the data files students will need.

LESSON STRUCTURE

DISCOVER

LESSON 1	In class model a Traffic light using Scratch.
Starter	Students individually complete Resource 1.1 Every journey matters (page 23).
Main 1	Use Resource 1.2 History of first gas traffic light in London (page 25) to establish a simple algorithm for traffic lights.
Main 2	Students produce an algorithm for traffic lights by one of Pseudocode, Flow chart or using Resource 1.4 algorithm cards (page 27).
Main 3	Students modify a model traffic light in Scratch using Resource 1.5 Simulation program (available as files to download).
Plenary	Assess progress through discussion of model effectiveness.
LESSON 1	In class model a Pedestrian Crossing using Scratch.
Starter	Whole class view video(s) to show that Transport for London are continually advancing traffic sensing and control technology in London.
Main 1	Group activity to establish an algorithm for a crossing with an input, using the following resources: 2.1 Pedestrian crossing unplugged (page 44) 2.2 Green man / red man crossing cards (page 45) 2.3 Algorithm cards (page 46)
Main 2	Students modify a model pedestrian crossing in Scratch using Resource 2.4 Simulation program (page 50).
Extension Challenge	Students modify an advanced model pedestrian crossing in Scratch using Resource 2.6 Advanced Simulation program (available as files to download).
Plenary	Assess progress through discussion of detectors and models of crossings.

LESSON STRUCTURE CONTINUED

LESSON 3	Access live London bus data from Transport for London (TfL) data.
Main 1	Teacher introduces basic Python.
Main 2	Students write own Python from given material.
Main 3	Students write own Python code to learn about Integer and String Objects and food menus to learn about Lists.
Main 4	Students write own Python code using London Bus data to learn about Dictionaries.
Main 5	Assess progress through discussion of written code.
Main 6	More coding of bus data.
Plenary	Assess progress through discussion of model effectiveness.
LESSON 4	Write own Python code to read and analyse real TfL data.
Main 1	Paired discussion on Transport for London TfL data not available publicly.
Main 2	Students obtain online data from TfL regarding Bike points without coding.
Main 3	Students write own code in Python for a simple loop.
Main 4	Students write Python code to read and display Bike data from TfL online data.
Extension Challenge	Write Python code to access Bus data.
LESSON 5	School Trip to a centre of transport, e.g. Main Train Station.
LESSON 6	Reflection and Evaluation of learning through trip in the classroom.
Starter	Teacher explains lesson.
Main 1	Students Identify a London transport problem.
Main 2	Students research and produce a poster, presentation or report.
Plenary	Present presentations.

EXPLORE

CONNECT

UNIT NOTES

Requirements to teach the unit

Teachers and students who have used Scratch and Python before should find this unit straight forward to follow. Those with very little or no experience will find some sections a challenge and may benefit from some extra time covering the required basics.

Python version 3.4 or later will be needed by students and teacher for lessons 3 and 4. Student and teacher Internet access is needed in all lessons to run Scratch 2.0 Online in lessons 1 (page 9) and 2 (page 28), and obtain London data in lesson 4 (page 69).

Internet access is also required to view various YouTube videos, mainly by teachers only.

At the start of each lesson there is specific information to help the teacher.

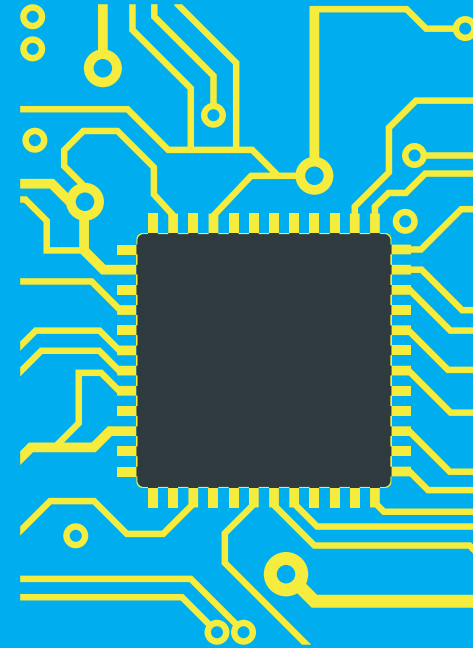


Les Chatfield 2014, Flickr

DISCOVER

Students will explore the different forms of transport that are used day to day in London and explore some of the challenges of ensuring that travel is reliable and safe as the population grows, measures to tackle pollution are introduced and cycling rates increase.

The Discover lessons will focus on the use of traffic lights to control road traffic in London, with programming examples used to simulate and explore different situations. The lessons start with the simplest model of a traffic light sequence repeating for a fixed time. By understanding the limitations of fixed timing for changing traffic conditions, students will then see the need for input sensors or detectors to sense traffic flow and also pedestrians and cyclists and use these inputs to control the traffic light sequence. The work on individual traffic lights will be extended to look at the larger problems involved in coordinating groups of traffic lights in areas of London and at how engineers ensure reliability of these systems and data.



LESSON 1

CONTROLLING LONDON'S TRANSPORT



THE BIG IDEA

Journeys in London for commuters, residents, tourists and anyone needing to travel between locations are vital to the life of our city. There are more forms of transport in London than students might imagine and it is a major part of the role of Transport for London to make journeys on all forms of transport work together reliably, efficiently and safely. Traffic lights provide a key method to manage road traffic flow.

Students will model a simple traffic light operation, develop an algorithm and work on a program to implement this operation.



LEARNING OUTCOMES

Students:

Could evaluate how an algorithm performs under changing conditions such as increased traffic flow

Could explain why reliability of journey times is important to travellers

Should describe a traffic light fixed time sequence algorithm and recognise this in a program

Must describe different forms of transport used in London and the number of journeys made each day



RESOURCES

Resource 1.1: Every journey matters

Resource 1.2: The first traffic light

Resource 1.3: Teacher instructions for algorithm cards

Resource 1.4: Student algorithm cards

Resource 1.5: Traffic light simulation programs – *available as files to download*

To facilitate differentiation, 3 models are provided for teachers to allocate one to each of their students:

1_5a_TrafficLight_Complete.sb2
completely coded

1_5a_TrafficLight_Partial.sb2
partially coded

1_5a_TrafficLight_Base.sb2 base model

LESSON 1

TRANSPORT IN LONDON AND LONDON TRANSPORT



KEYWORDS

- ◆ Computational thinking
- ◆ Algorithm
- ◆ Abstraction
- ◆ Sequence
- ◆ Repetition

EXTENSION KEYWORDS

- ◆ Frequency
- ◆ Reliability

EXTERNAL LINKS

TfL youtube channel:

www.youtube.com/channel/UCmh0HDSxGTWjcll5mcfEhGA

Every journey matters youtube video
(2 mins 44 sec):

www.youtube.com/watch?v=J-JpmFDeBo

Computational thinking, a guide for
teachers:

community.computingatschool.org.uk/resources/2324

Wikipedia traffic light article:

en.wikipedia.org/wiki/Traffic_light

London's Transport a history:

tfl.gov.uk/corporate/about-tfl/culture-and-heritage/londons-transport-a-history

LESSON 1: CONTROLLING LONDON'S TRANSPORT

SETTING THE SCENE

Keeping London on the move

Students will have experienced common forms of transport depending on the area of London they live in. Most will have experienced the bus, cars and the underground but fewer will have used trams and river transport.

In addition to the different types of transport used in London, students should be made aware of the high number of journeys on any typical day and the need for these journeys to be 'reliable' in terms of timing. A simple explanation of reliability can be expressed as consistency of journey time. If a journey to school or place of work takes 30 minutes then if transport is reliable the time will be close to 30 minutes on most days, it will not often be 15 minutes or 45 minutes.

The complexity of journeys in London cannot be managed by methods that always work in the same way. Transport systems collect data through a wide range of detectors over time for longer term planning but also in real-time for making instantaneous decisions. These might be decisions made by the computer programs controlling one or more traffic lights or decisions made by a person seeing that there is a long delay for the next bus.

Changing conditions in London also require transport systems to adapt. An increasing population, the need to reduce congestion and pollution, reduced use of private transport and increased use of public transport and cycling are all factors being taken into account in London transport. Measures are also taken to promote some forms of transport over others, such as public transport in preference to private transport.



ROAD TRAFFIC IN WHITEHALL
© Koïs Miah

LESSON 1: CONTROLLING LONDON'S TRANSPORT

SETTING THE SCENE

The role of computational thinking

This unit focuses on road traffic signals in London and how they are used to manage travel by road. This includes buses and cars but also pedestrians and cyclists. After a general introduction to transport in London students are introduced to an 'unplugged' model of a pedestrian crossing signal.

Understanding the unplugged model leads to a computer model, a program, which simulates the traffic signal. The algorithmic thinking in the examples used starts with a simple traffic light model that sequences for a fixed time period, for example repeating the cycle every one minute with a green light for 30 seconds. Evaluations of this will show that this is a poor solution when traffic is very busy or very light and this will lead students to the idea of sensing the traffic and developing algorithms for better solutions.

Algorithm design and evaluation of algorithms are both aspects of problem solving through computational thinking.

Another aspect of computational thinking applied in this case is abstraction – filtering out the details that are not important to the problem. In this case light, colour, sequence and timing are considered. Other details such as the size and height of the lights, the type of bulb used are ignored or hidden. These details are not relevant to this computational solution but they will be important for other tasks such as building and positioning of the lights.

The article *Developing computational thinking in the classroom: a framework* is recommended for information and activities to develop Computational thinking in the classroom.

<http://academy.bcs.org/sites/academy.bcs.org/files/DevelopingComputationalThinkingInTheClassroomaFramework.pdf>

LESSON 1: CONTROLLING LONDON'S TRANSPORT

ACTIVITIES

STARTER

To introduce transport in London.

Give the students Resource 1.1 Every journey matters (page 23) and explain that they are to make notes of the different forms of transport in London and the uses of computing or technology in the journeys, from the Transport for London *Every journey matters* video. Length: 2m 44s

www.youtube.com/channel/UCmh0HDSxGTWjcll5mcfEhGA

A first example is given: Buses use the live bus arrival time displays on many stops now and also the Oyster card rather than cash payment.

Other forms of transport shown or mentioned in the video are Tube.

Discussion questions

- ◆ How does information help journeys? For example, how does the display of live bus arrival times help passengers and the transport system?
- ◆ What is meant by reliability of journey?

Resource 1.1 Every journey matters (page 23) may be used as a final assessment by asking students about the uses of technology and computing in a chosen aspect of the transport system at the end of the unit and comparing to the answers given at this stage.

MAIN 1

To introduce a simple algorithm for the traffic light sequence.

Ask students to read Resource 1.2 History of first gas traffic light in London (page 25) to set the context and discuss what they understand by 'manual' and 'automatic' traffic light operation.

Check that students know the normal sequence for traffic lights: red, red/amber, green and amber repeated automatically. Check also that they know that the traffic light would have to be red, to stop traffic, whilst pedestrians cross.

Discussion questions

- ◆ What are the disadvantages of a fixed time sequence for the traffic lights?
Hint: think or act out conditions where there are no pedestrians, many pedestrians, heavy traffic etc.

MAIN 2: ALGORITHM DESIGN

An algorithm is a clear set of steps used to solve a problem. Computational thinking requires the problem to be analysed so that these steps can be defined. For a small problem such as the traffic light sequence, students should be able to write down an algorithm either in a simple text form (pseudocode) or as a flowchart, before coding in a programming language.

Ensure that the students have recorded the steps in their method. You may use any flowchart or pseudocode method that students are familiar with or use Resource 1.4: Student Algorithm cards (page 25). If you use your own methods, you should still produce a similar algorithm.

Differentiation

Give lower ability students a blank framework showing them the steps in sequence and the loop.

MAIN 3: PROGRAMMING – IMPLEMENTING THE ALGORITHM

Students work through the example programs and modification tasks.

Differentiation in programming tasks – lower ability students may be given complete programs to make minor modifications or work with higher ability students. High ability students may be given partially functioning programs that require changes to the program structure to complete.

Follow the guidance Resource 1.5 Traffic light simulation programs (available as files to download) to complete the tasks and for ideas about how to develop student skills in coding, code reading and debugging.

Ensure that:

- ◆ Students compare their predictions and final solutions to their algorithm design
- ◆ Students have recognised sequence and repetition structures and instructions in both algorithm and program code.

PLENARY

For teachers there are two suggestions here depending on available time. If you are short on time, then consider a busy junction near the school that students are familiar with. If you have plenty of time, then watch the video clip *Peter on traffic impact modelling* on the TfL roads page:

tfl.gov.uk/roads

or you can use the direct youtube link

**[www.youtube.com/
watch?v=JBzhr1T3ADY](https://www.youtube.com/watch?v=JBzhr1T3ADY)**

which shows modelling of the traffic flow around Westminster.

Question:

What similarities and what differences did you notice between Peter's models and our own unplugged model?

Answer:

It is similar in the sense that he is using a model of the real situation to analyse and solve the problem.

LESSON 1: CONTROLLING LONDON'S TRANSPORT

FURTHER ACTIVITIES

Assessment questions

Adapt or extend questions based on the activities or use the following examples:

- ◆ Explain with an example what is meant by the term 'Algorithm'.
- ◆ Give examples of program instructions used in 'Sequence' and in 'Repetition' from the traffic light program.
- ◆ Abstraction is a computational thinking method that considers the variables important to solving the problem and ignores others. In this example the colours of the traffic lights are important but the size of the lights is not. List the other variables that are considered important in these programs and some others that are not important.

Homework ideas

- ◆ Research the London Congestion Charge. Describe reasons for introducing this in London and the computing technology needed.
- ◆ Using the London's Transport a History weblink
<https://tfl.gov.uk/corporate/about-tfl/culture-and-heritage/londons-transport-a-history>
- ◆ Allocate groups of students a different form of transport: tube (choose a local line), river, bus, coach, tram, London Overground or DLR. Ask each group to make a one minute presentation on their form of transport. Other sources of information are available at:
www.20thcenturylondon.org.uk/theme/transport
www.ltmcollection.org/museum/index.html
- ◆ Using the example Scratch or Python program as a model, adapt or write a new program for a different timed sequence e.g. a clock, day and night, plant growing.

Further reading:

If you are a member of Computing At School you can read the article on computational thinking from Computing at Schools page:

<http://community.computingatschool.org.uk/resources/252>

LESSON 1: CONTROLLING LONDON'S TRANSPORT

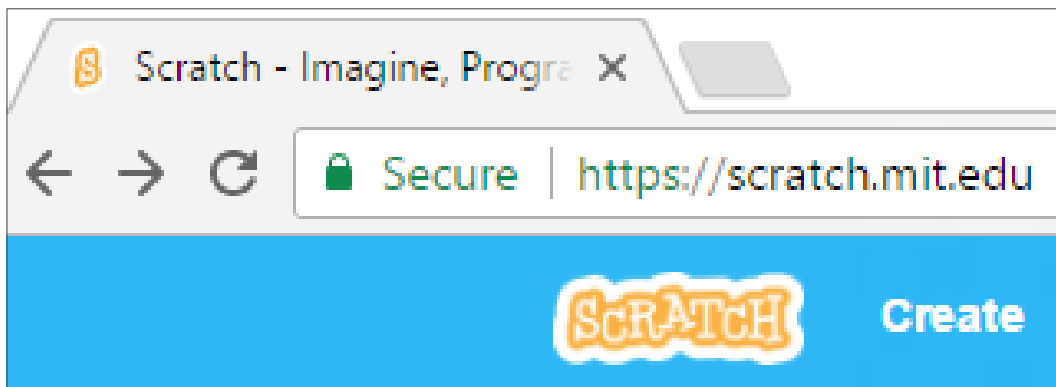


STUDENT SCRATCH ACTIVITIES

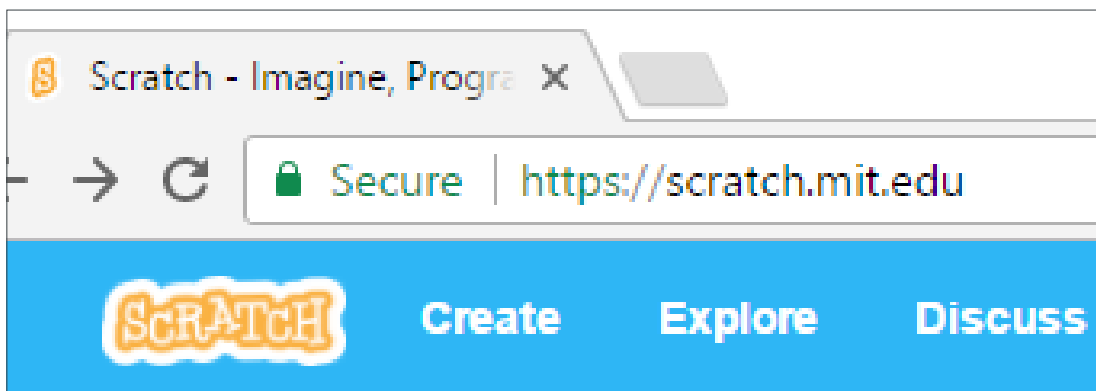
We're going to be using Scratch version 2 (If your teacher would prefer you to use a locally installed Scratch program you can use that and skip the first 2 steps).

Scratch is readily available online and you can access it as follows:

1.1 Enter **scratch.mit.edu/** into your browser as shown below:



1.2 From the Scratch website home page click on **Create**:

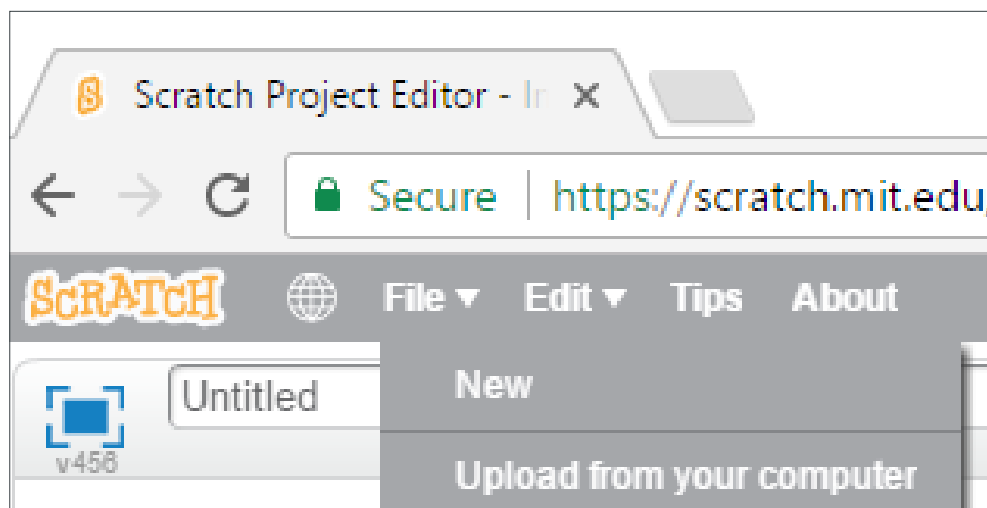


LESSON 1: CONTROLLING LONDON'S TRANSPORT

STUDENT SCRATCH ACTIVITIES CONTINUED



1.3 From the Scratch Create screen select **File, Upload from your computer**:



1.4 Carefully navigate to where your teacher has stored the traffic light program they wish you to use, select it and then press **Open**:

Name	Date modified	Type	Size
Connected City 2017	15/05/2017 14:21	File folder	
1_5a_TrafficLight_Complete.sb2	20/09/2016 11:00	SB2 File	13 KB
1_5b_TrafficLight_Partial.sb2	20/09/2016 11:00	SB2 File	11 KB
1_5c_TrafficLight_Base.sb2	20/09/2016 11:00	SB2 File	11 KB
2_4a_TrafficLightInterrupt_Complete.sb2	20/09/2016 11:00	SB2 File	18 KB
2_4b_TrafficLightInterrupt_Partial.sb2	20/09/2016 11:00	SB2 File	18 KB
2_6_Advanced_TrafficLightInterrupt_Com...	20/09/2016 11:00	SB2 File	20 KB
3_1a_VariablesVsList.sb2	20/09/2016 11:00	SB2 File	55 KB

Note for teachers

This will be one of:

1_5a_TrafficLight_Complete.sb2 As shown below 1.5 to 1.10

1_5a_TrafficLight_Partial.sb2 As shown below 1.11 to 1.14

1_5a_TrafficLight_Base.sb2 As shown below 1.15 to 1.18

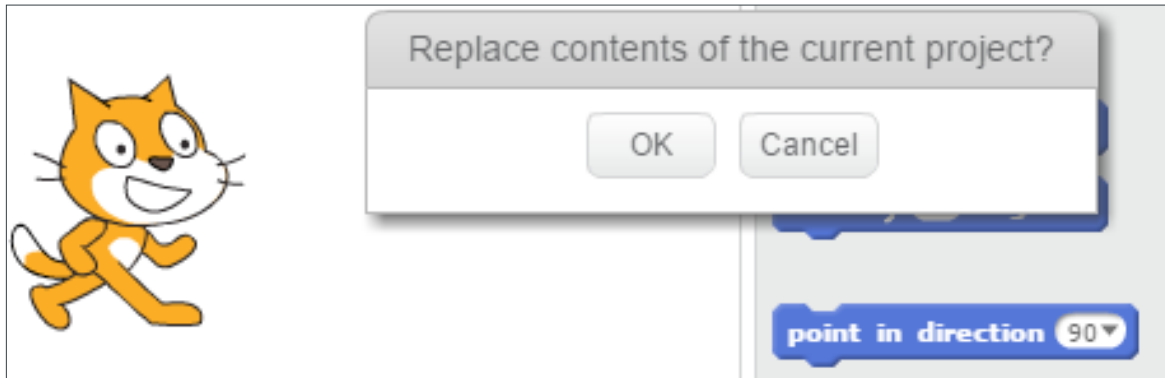
If using Scratch 1.4, you will need to load the Sample programs using Scratch 2 and save 1.4 versions for your students.

LESSON 1: CONTROLLING LONDON'S TRANSPORT

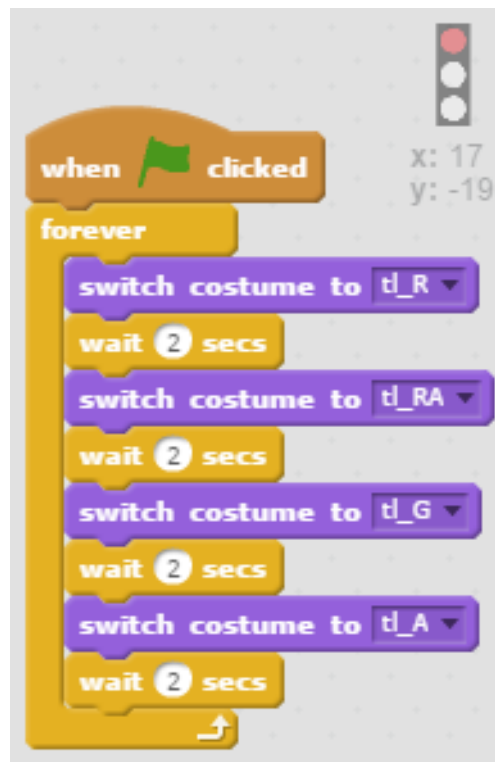
SCRATCH COMPLETE ACTIVITIES



1.5 Click on **OK** to **Replace contents of the current project?**



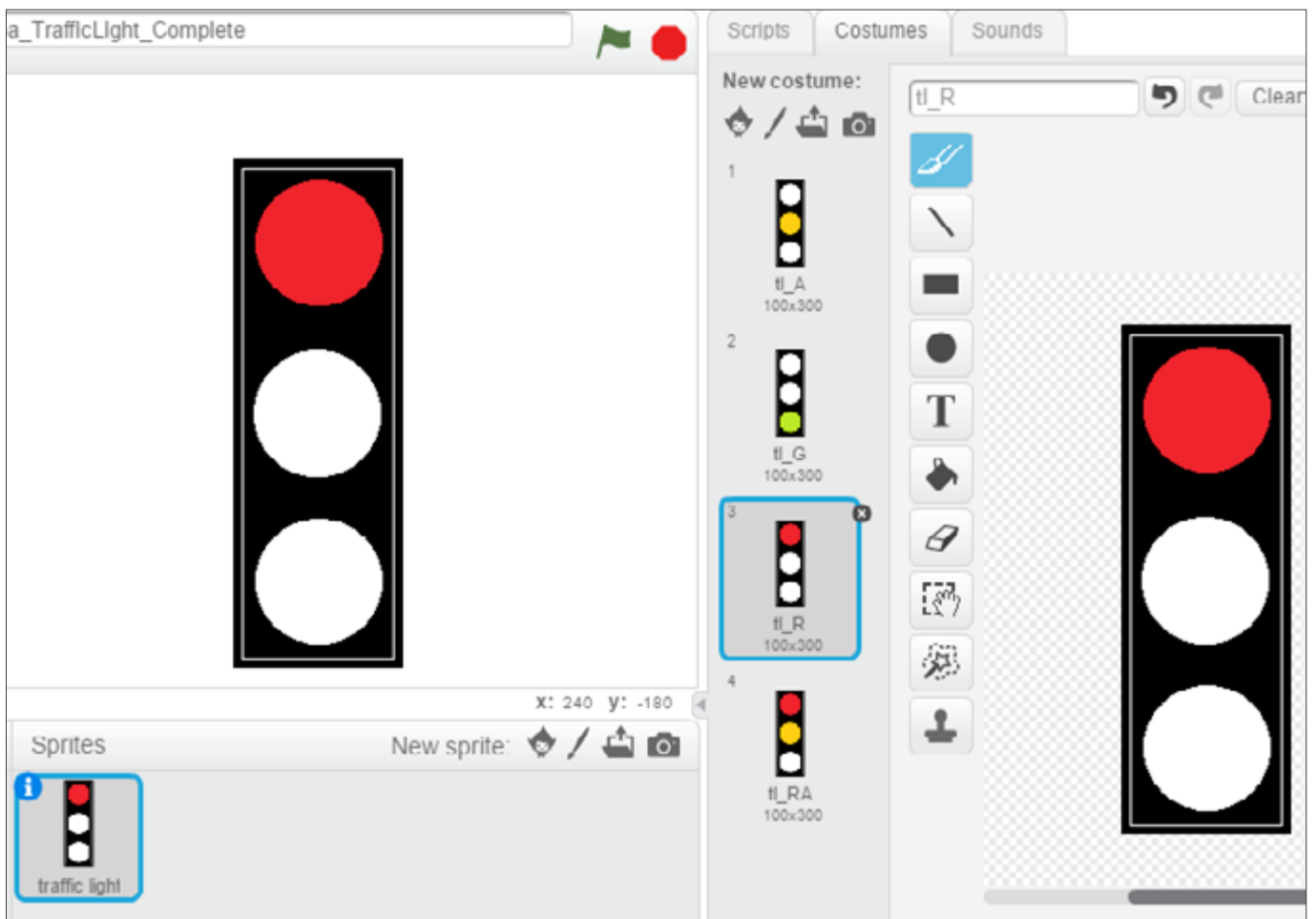
1.6 Read through the code and predict what you think the program will do BEFORE running the program:



LESSON 1: CONTROLLING LONDON'S TRANSPORT SCRATCH COMPLETE ACTIVITIES CONTINUED



1.7 You might find it useful to look at the different Costumes for the Traffic light sprite, by selecting **Costumes**:

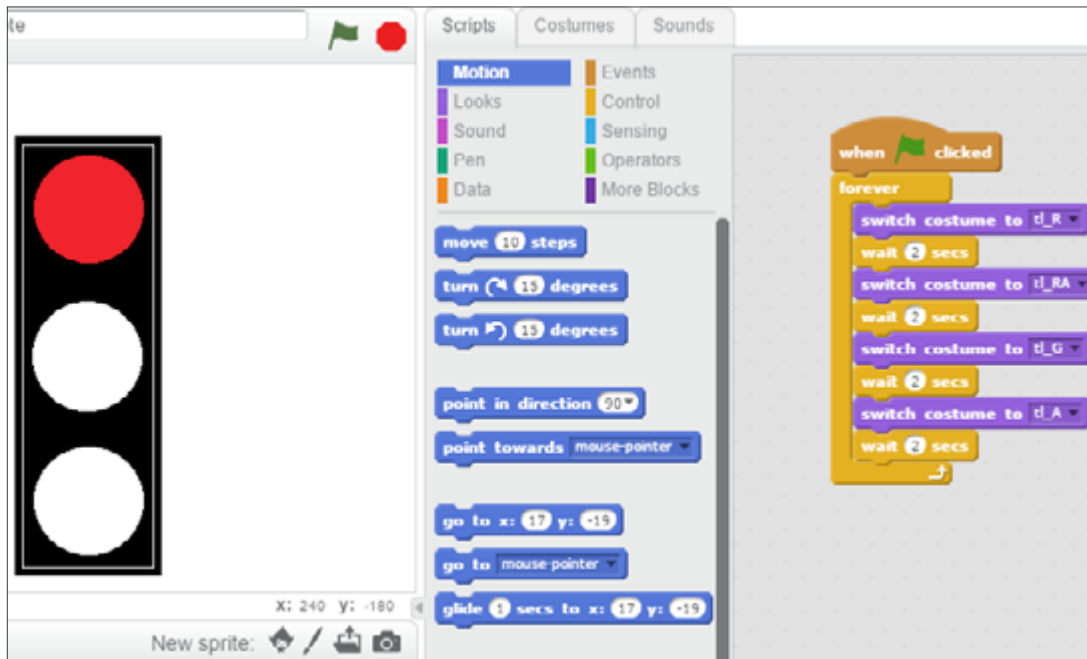


LESSON 1: CONTROLLING LONDON'S TRANSPORT

STUDENT SCRATCH ACTIVITIES CONTINUED

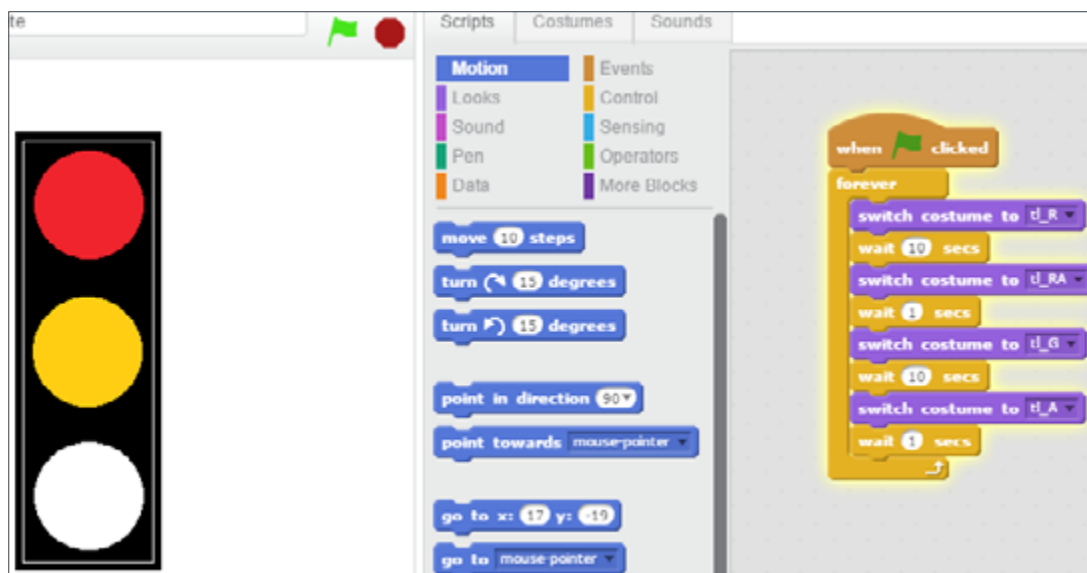


1.8 Select the **green flag** to run the program and confirm whether your prediction was correct or not. If not, try to explain, which part of the code has behaved differently from what you predicted.



1.9 Ensure the Script is visible by **Selecting Scripts** if you are showing costumes and modify the program to work in the same way as the algorithm you designed earlier in the lesson. That is change the wait times and or sequence of colours.

1.10 As an example you may get:

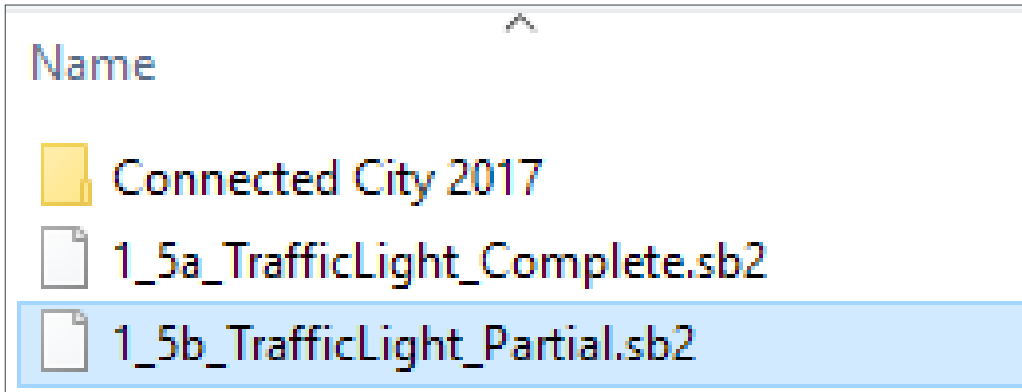


LESSON 1: CONTROLLING LONDON'S TRANSPORT

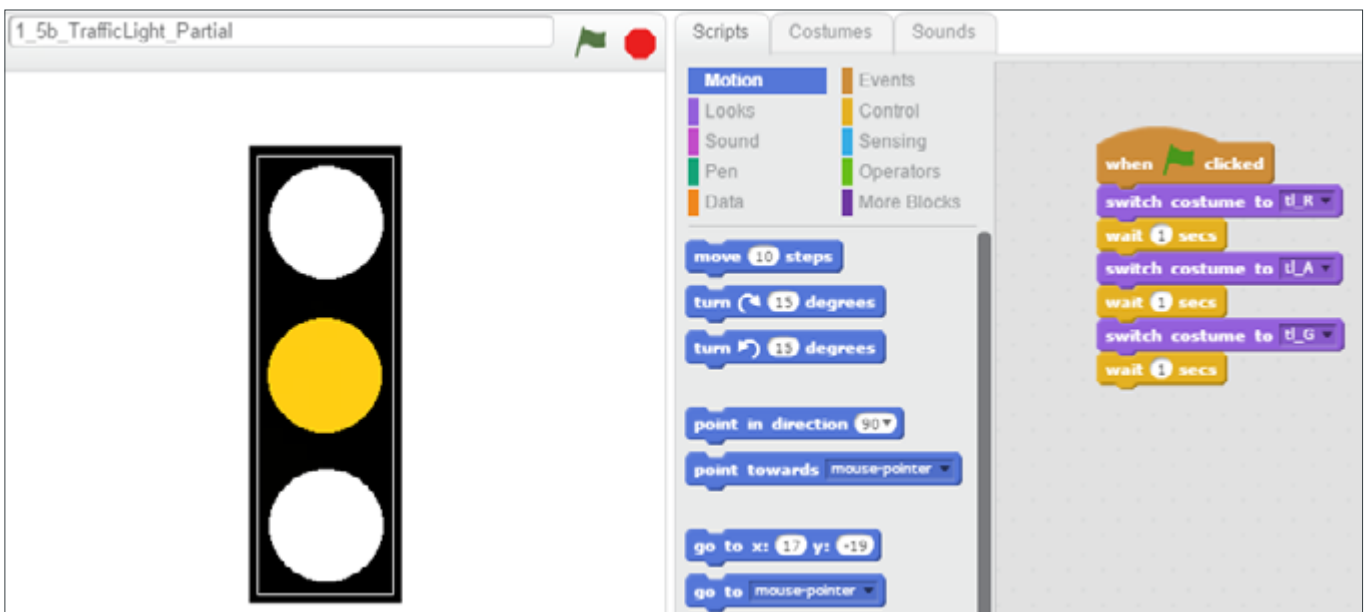
SCRATCH PARTIAL ACTIVITIES



1.11



1.12 Read through the code and predict what you think the program will do BEFORE running the program.



1.13 Select the **green flag** to run the program and confirm whether your prediction was correct or not. If not, try to explain, which part of the code has behaved differently from what you predicted.

1.14 Modify the program to work in the same way as the algorithm you designed earlier in the lesson. That is change the wait times and or sequence of colours.

Challenge

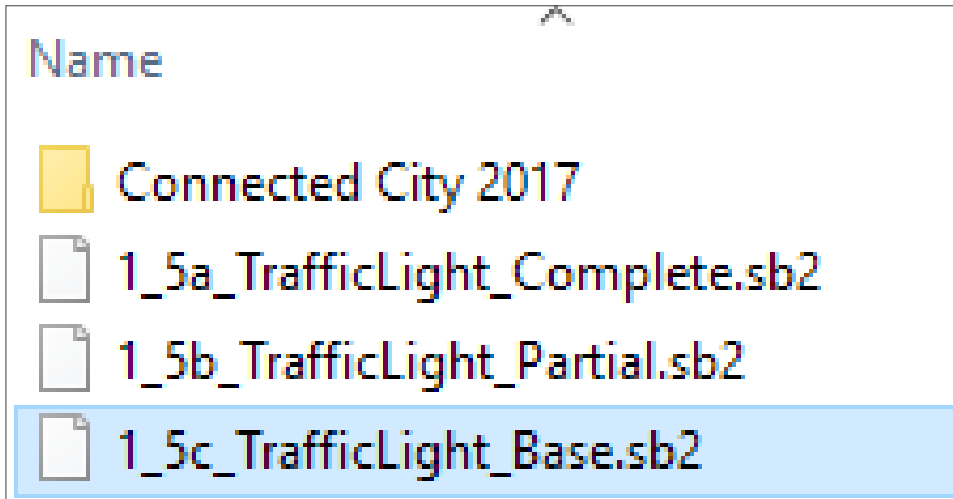
Add a new costume for Red and Amber and adjust the code to use it.

LESSON 1: CONTROLLING LONDON'S TRANSPORT

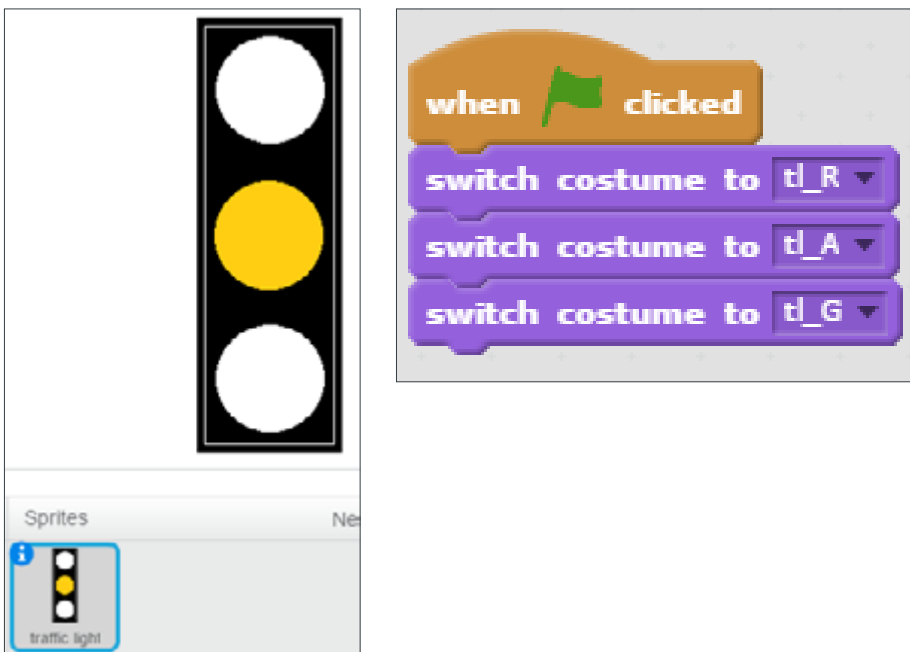
SCRATCH BASE ACTIVITIES



1.15



1.16 Read through the code and predict what you think the program will do BEFORE running the program.



1.17 Select the **green flag** to run the program and confirm whether your prediction was correct or not. If not, try to explain, which part of

1.18 Modify the program to work in the same way as the algorithm you designed earlier in the lesson. That is change the wait times and or sequence of colours.

Challenge

Add a new costume for Red and Amber and adjust the code to use it.

LESSON 1: TRANSPORT IN LONDON AND LONDON TRANSPORT

RESOURCE 1.1: EVERY JOURNEY MATTERS



Watch the *Every Journey Matters* video

www.youtube.com/channel/UCmh0HDSxGTWjcll5mcfEhGA

and make a note of the different forms of transport in London that you see and any uses of technology or computing in each.

Form of transport	Use of technology/computing
Bus	Live bus arrival information at stops Oyster card

LESSON 1: TRANSPORT IN LONDON AND LONDON TRANSPORT



RESOURCE 1.1: EVERY JOURNEY MATTERS CONTINUED

The video talks about improving **frequency** and **reliability** of buses and other transport. Discuss the difference between these two terms in your group then write a short definition for each.

Term	Definition
Frequency	
Reliability	

LESSON 1: TRANSPORT IN LONDON AND LONDON TRANSPORT

RESOURCE 1.2: THE FIRST TRAFFIC LIGHT



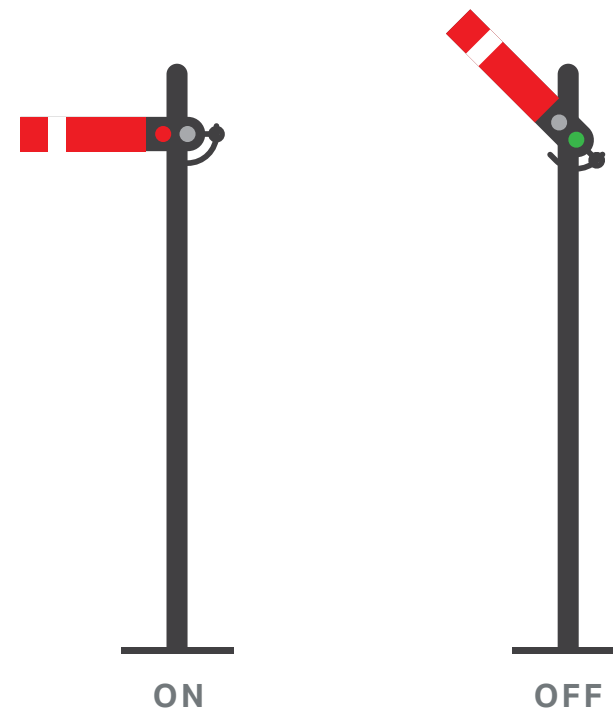
From en.wikipedia.org/wiki/Traffic_light

On 9 December 1868 the first traffic lights in Britain were installed outside the Houses of Parliament in London to control the traffic. The designer took his inspiration from railway signals of the day. His design combined three semaphore arms with red and green gas lamps for night-time use, on a pillar, operated by a police constable. The gas lantern was manually turned by a traffic police officer, with a lever at its base so that the right light faced the traffic.

Although history tells us that it was successful in controlling the traffic, this technology did not last long because the lights exploded early the following year injuring or killing the policeman who was operating them. This system was abandoned until electric signals became available. The first electric traffic lights in England were installed in Piccadilly Circus in 1926.

Students might discuss other forms of technology that seemed a brilliant idea but were abandoned for various reasons.

Can you spot semaphore signals like these still in use on our railways?



LESSON 1: TRANSPORT IN LONDON AND LONDON TRANSPORT

RESOURCE 1.3: TEACHER INSTRUCTIONS FOR ALGORITHM CARDS



Teacher instructions

Cut out the cards on the next page and ask students in groups to sort these out to represent the traffic light operation algorithm, in the framework provided.

A sample sequence and timings are suggested here. Students should produce their own timings and justify that they provide a safe crossing for pedestrians.


Repeat:	
	Green light
	Wait 30 seconds
	Amber light
	Wait 5 seconds
	Red light
	Wait 10 seconds
	Red + amber light
	Wait 5 seconds


LESSON 1: TRANSPORT IN LONDON AND LONDON TRANSPORT

RESOURCE 1.4: STUDENT ALGORITHM CARDS



Blank framework:

 Repeat
Light colour(s)
Light colour(s)
Light colour(s)
Light colour(s)
Light colour(s)
WAIT seconds
WAIT seconds
WAIT seconds
WAIT seconds
WAIT seconds

 Repeat
Light colour(s)
Light colour(s)
Light colour(s)
Light colour(s)
Light colour(s)
WAIT seconds
WAIT seconds
WAIT seconds
WAIT seconds
WAIT seconds

LESSON 2

TRAFFIC SENSING TECHNOLOGY

SETTING THE SCENE

The need for traffic control to be responsive to changing conditions has been recognised for many years. Before the days of automated devices signalling would have been controlled manually. As technology advances more and better ways of collecting data about traffic and conditions on the road have also developed and are continuously being integrated into London's transport system.

This technology has developed in answer to the needs of the city. The changing nature and volume of transport on London's roads requires more complex control systems and this has been made possible by new technology development in hardware, software and communications.

Only five years ago the majority of engineers working on traffic lights would be involved in making direct modifications to the lights themselves, but now the major requirement is for engineers to work with computers and code programs to monitor and control these systems.



BYCICLE TRAFFIC SIGNALS

© Transport for London

LESSON 2

TRAFFIC SENSING TECHNOLOGY



THE BIG IDEA

The types and numbers of travellers vary hugely at different times of day on London's roads, meaning that simple fixed algorithms that do not respond to real conditions would be very inefficient. Students will further develop their model of the pedestrian crossing, to operate conditionally in response to pedestrians pressing the crossing button rather than in a fixed time sequence.



LEARNING OUTCOMES

Students:

Could repeatedly use and evaluate a model of a traffic light with interruption

Should describe a traffic light sequence algorithm which reacts to input from pedestrian crossing button

Must use Scratch programming language to model a traffic light with interruption



RESOURCES

Resource 2.1: Pedestrian crossing unplugged (with interruption)

Resource 2.2: Green man / red man crossing cards

Resource 2.3: Car and traffic light cards

Resource 2.4: Algorithm cards

Resource 2.6: Traffic light with interruption – *available as files to download*

Stop watches or student phones for timing

LESSON 2

TRAFFIC SENSING TECHNOLOGY



KEYWORDS

- ◆ Selection
- ◆ Condition
- ◆ Input

EXTERNAL LINKS

London Transport Museum Films page:
www.ltmcollection.org/films/index.html

Transport for London Roads Plan:
www.tfl.gov.uk/roads

Transport for London's news article on
new technology being trialled to
detect cyclists:

tfl.gov.uk/info-for/media/press-releases/2015/june/new-pioneering-trials-to-detect-cyclists-at-junctions-begin-in-london

LESSON 2: TRAFFIC SENSING TECHNOLOGY

ACTIVITIES

STARTER

To show that Transport for London are continually advancing traffic sensing and control technology in London.

View Chapter 5 of the *London on the Move* video (On 1970s sensing devices for London buses and the manual decision making involved) from the London Transport Museum Film Collection at:

www.ltmcollection.org/films/index.html

Then view *The people behind London's roads - Andrew Wiseall* video from the Transport for London Roads page:

tfl.gov.uk/roads

(Showing a team using laptops to link to traffic lights and monitor the data inputs and how the light responds.)

MAIN 1

To introduce an algorithm for interrupting a traffic light sequence by pressing a button at a pedestrian crossing.

Organise students into groups and hand out Resource 2.1 Pedestrian crossing unplugged (with interruption) (page 44) and Resource 2.2 Green man/red man crossing cards (page 45). Students will also need the traffic light and car cards from Resource 2.3 Pedestrian crossing unplugged (page 44). Carry out the unplugged activity with students acting as traffic lights and timer for the automatic operation of traffic lights as before but with one of the student pedestrians operating the crossing to interrupt the traffic light and one student acting as the green man/red man pedestrian light.

In this model the traffic light is now continuously green to allow flow of road traffic but can be interrupted by a press on the pedestrian crossing button which then runs through the timed sequence of lights. Students should recognise this as

a realistic model of a pedestrian crossing. Explain that the program is very similar to the fixed time sequence program except that it now has a conditional (or selection) element, in that the part of the program that goes through the amber and red sequence of lights is only carried out if the button is pressed. The program is conditional on sensing a real world event.

Question

What other events might interrupt or affect the traffic light sequence on normal roads?

Answers

Buses given priority on bus lane, cycles on cycle lane, heavier flow traffic or number of pedestrians.

The key point to elicit is that in each case there needs to be one or more sensors to sense the event and the algorithm and program will take account of this sensor input. The button on the pedestrian crossing is the 'sensor' that inputs a pedestrian wanting to cross event.

LESSON 2: TRAFFIC SENSING TECHNOLOGY

MAIN 2

Algorithm design.

Again, having modelled the steps unplugged, students should now write down their algorithm in their chosen format or use Resource 2.4: Algorithm cards (page 50).

The key change from the previous lesson is that this will involve adding a selection structure to the algorithm and program.

Many alternative solutions are possible and will vary for each type of junction and light. For this scenario the behaviour required is continuous green interrupted with the normal sequence when the pedestrian crossing button is pressed.

Differentiation in algorithm task – give lower ability students a blank framework showing them the steps in sequence and the loop.

MAIN 3

Programming – implementing the algorithm (examples in Scratch)

Example programs for a traffic light with pedestrian crossing with differentiated modification tasks.

Students complete the program modification task examples for this lesson in a similar way to lesson 1.

Make sure that students have recognised the selection **press c from the keyboard** instruction and the condition being tested, **when c is pressed**.

Challenge Advanced program

This example uses more advanced programming techniques to display both the traffic lights and the pedestrian crossing green/red man display synchronised together. This is done in an object-oriented fashion with each sprite running its own code and messages (broadcasts) used to synchronise.

Higher ability students might be given the opportunity to explore this program in their own time or for homework.

LESSON 2: TRAFFIC SENSING TECHNOLOGY

PLENARY

Ask students what should happen if more than one detector is sensing events at the same time. For example, what should happen if a pedestrian presses the crossing button and a car or other vehicle is approaching at some speed?

Discuss the students' answers. There is no one correct answer. One response might be to give the pedestrian priority if the car is travelling sufficiently slowly to stop safely, otherwise to give the car priority. The engineer would have to decide what should be done and write the code accordingly then evaluate the solution to see how it worked and if it could be improved. Real programs require complex combinations of Boolean conditions to try to control real behaviour.

For example:

```
IF pedestrian waiting and car approaching THEN
    IF speed < 20mph THEN
        Start traffic light sequence for crossing
    ELSE
        Wait for car to pass
```

Adapt or extend questions based on the activities or use the following examples:

- ◆ Give examples of program instructions used in 'Selection' in traffic light program examples from this lesson.
- ◆ Write a program for a simple interactive greeting card which displays an initial image and then on pressing a key displays an animation by showing a short sequence of further images before returning to the initial image. Describe how the algorithm for this program is similar or different from the pedestrian crossing algorithm in this lesson.

LESSON 2: TRAFFIC SENSING TECHNOLOGY

Homework ideas

Programming – adapt the program to play a pedestrian crossing beep while the green man is on.

Research

Read the news article link below from Transport for London on developing the use of SCOOT to detect cyclists:

tfl.gov.uk/info-for/media/press-releases/2015/june/new-pioneering-trials-to-detect-cyclists-at-junctions-begin-in-london

Write a short summary of how SCOOT works and how our algorithms might be changed to incorporate this.



© Andy Wilson 2008, Flickr

LESSON 2: TRAFFIC SENSING TECHNOLOGY

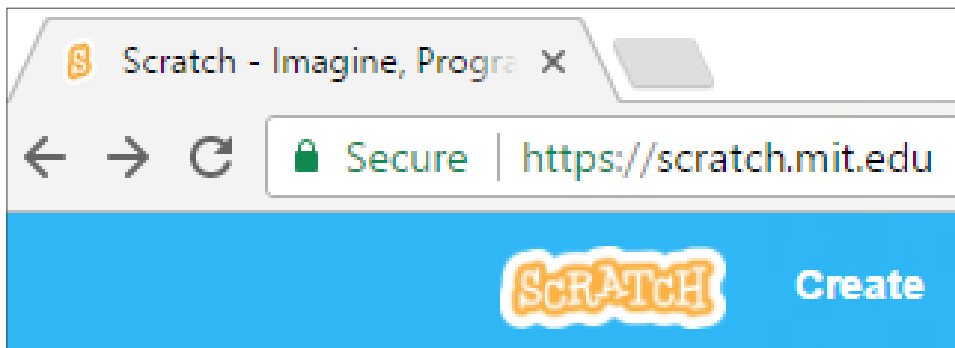


ACTIVITY 2.1: SCRATCH

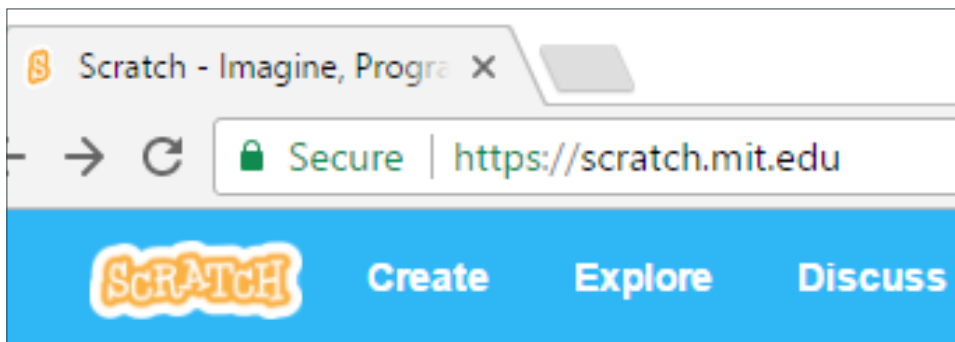
We're going to be using Scratch version 2 (If your teacher would prefer you to use a locally installed Scratch program you can use that and skip the first 2 steps).

Scratch is readily available online and you can access it as follows:

2.1 Enter **scratch.mit.edu/** into your browser as shown below:



2.2 From the Scratch website home page click on **Create**:

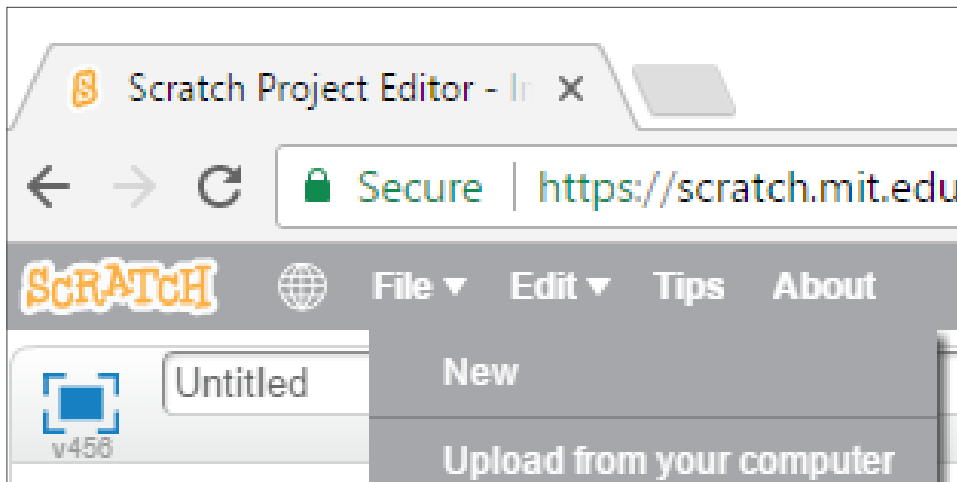


LESSON 2: TRAFFIC SENSING TECHNOLOGY



ACTIVITY 2.1: SCRATCH CONTINUED

2.3 From the Scratch Create screen select **File, Upload from your computer**:



2.4 Carefully navigate to where your teacher has stored the traffic light program they wish you to use, select it and then press **Open**:

Name	Date modified	Type	Size
Connected City 2017	15/05/2017 14:21	File folder	
1_5a_TrafficLight_Complete.sb2	20/09/2016 11:00	SB2 File	13 KB
1_5b_TrafficLight_Partial.sb2	20/09/2016 11:00	SB2 File	11 KB
1_5c_TrafficLight_Base.sb2	20/09/2016 11:00	SB2 File	11 KB
2_4a_TrafficLightInterrupt_Complete.sb2	20/09/2016 11:00	SB2 File	18 KB
2_4b_TrafficLightInterrupt_Partial.sb2	20/09/2016 11:00	SB2 File	18 KB
2_6_Advanced_TrafficLightInterrupt_Com...	20/09/2016 11:00	SB2 File	20 KB
3_1a_VariablesVsList.sb2	20/09/2016 11:00	SB2 File	55 KB

Note for teachers

This will be one of:

2.4a_TrafficLightInterupt_Complete.sb2

As shown below 2.5 to 2.10

2.4b_TrafficLight_Interupt_Partial.sb2

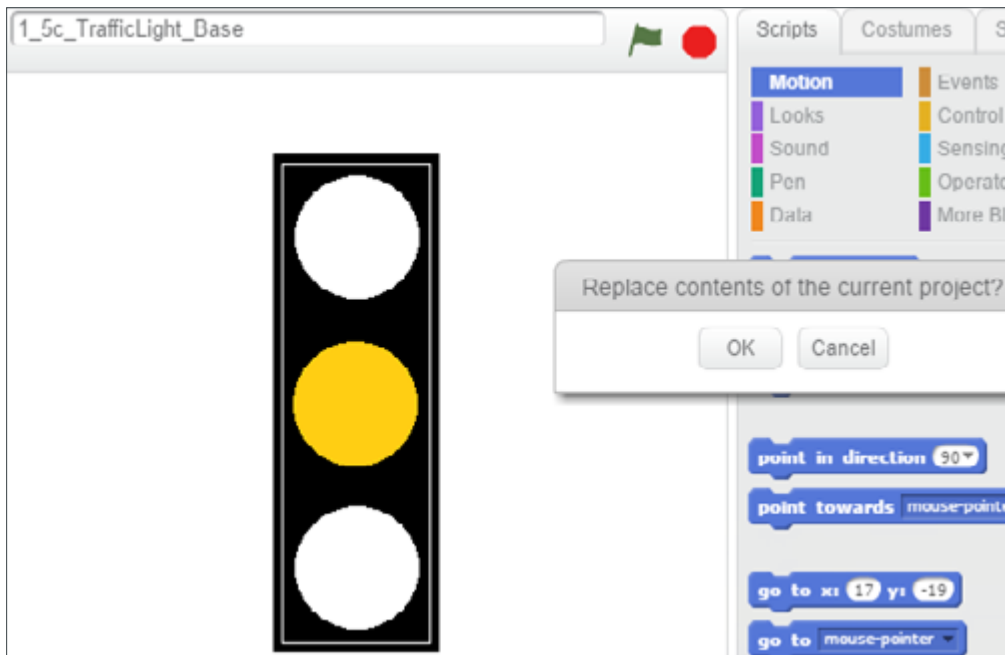
As shown below 2.11 to 2.14

LESSON 2: TRAFFIC SENSING TECHNOLOGY

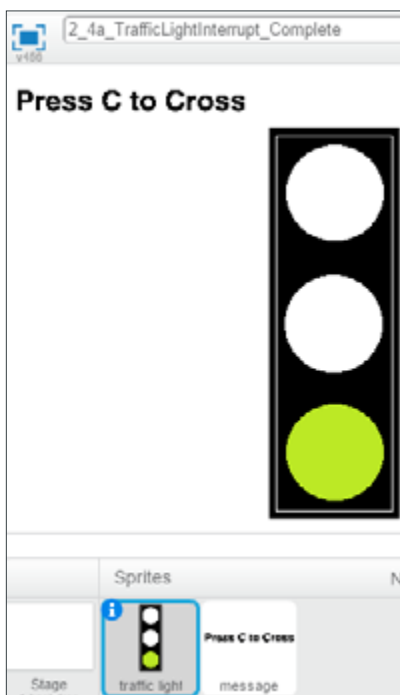
ACTIVITY 2.1: SCRATCH PARTIAL



2.5 Click on **OK** to Replace contents of the current project?



2.6 Read through the code and predict what you think the program will do BEFORE running the program.

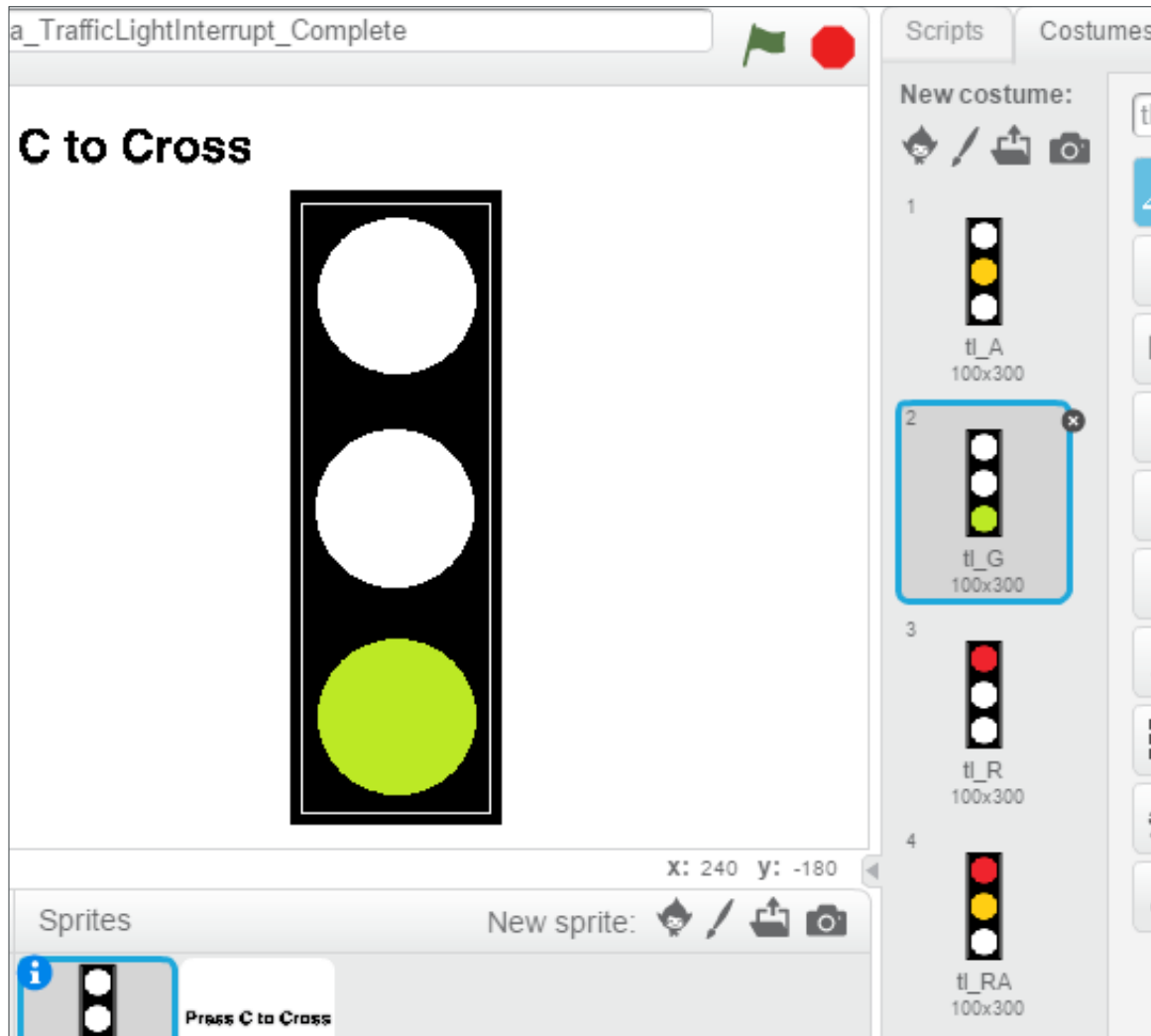


LESSON 2: TRAFFIC SENSING TECHNOLOGY

ACTIVITY 2.1: SCRATCH PARTIAL CONTINUED



2.7 You might find it useful to look at the different Costumes for the Traffic light sprite, by selecting **Costumes**



2.8 Select the **green flag** to run the program and PRESS C on the keyboard to cross. Confirm whether your prediction was correct or not. If not try to explain which part of the code has behaved differently from what you predicted.

2.9 Ensure the Script is visible by Selecting Scripts if you are showing costumes and modify the program to work in the same way as the algorithm you designed earlier in the lesson. That is change the wait times and or sequence of colours.

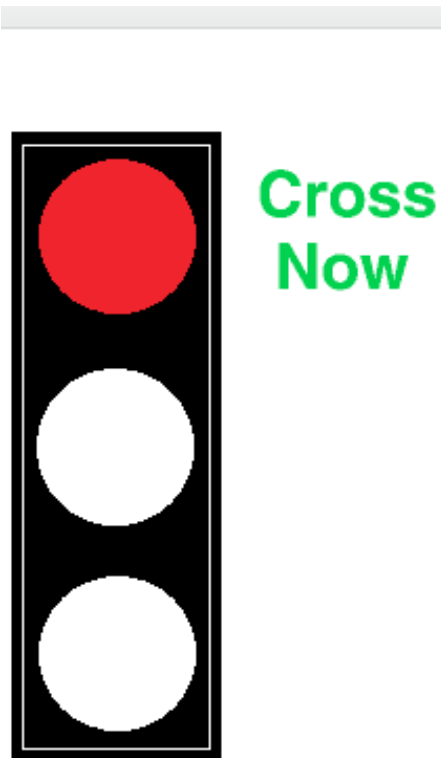
LESSON 2: TRAFFIC SENSING TECHNOLOGY

ACTIVITY 2.1: SCRATCH PARTIAL CONTINUED



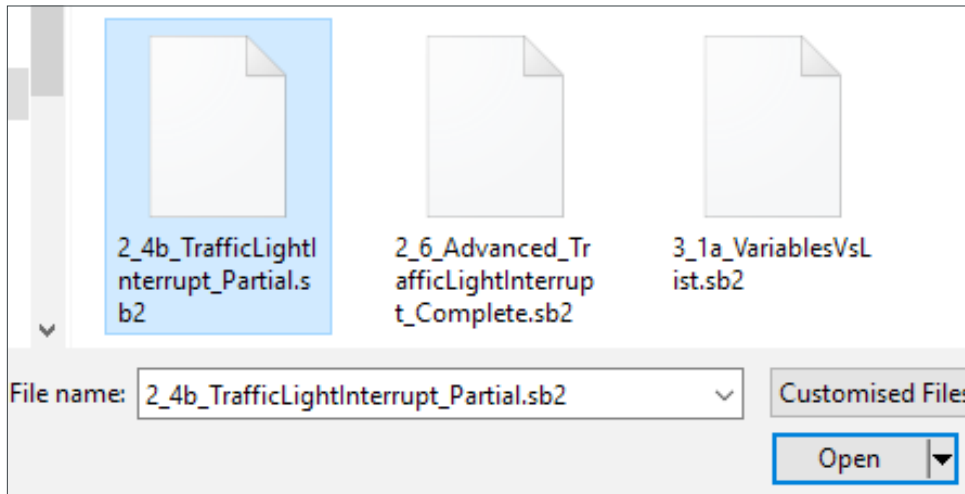
2:10 As an example, you may get:

```
when clicked
  forever
    switch costume to t1_G
    wait until key c pressed?
    wait 1 secs
    switch costume to t1_A
    wait 3 secs
    switch costume to t1_Rcross
    wait 4 secs
    switch costume to t1_RA
    wait 2 secs
```

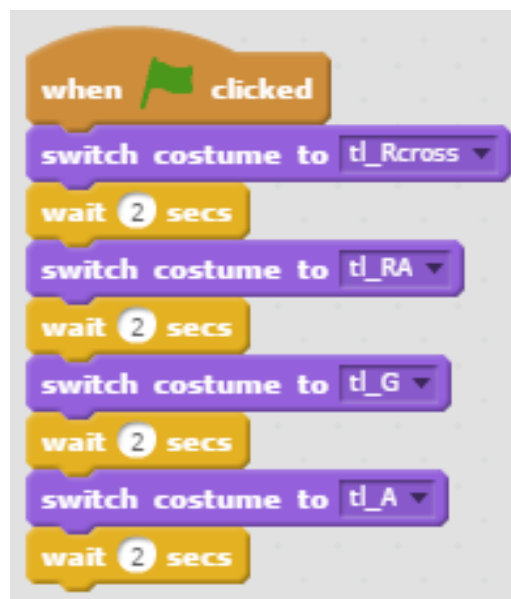
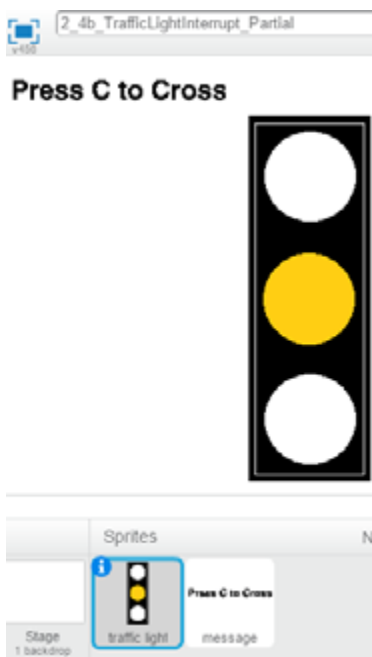


LESSON 2: TRAFFIC SENSING TECHNOLOGY

ACTIVITY 2.1: SCRATCH PARTIAL CONTINUED



2.11 Read through the code and predict what you think the program will do BEFORE running the program:



2.12 Select the Green flag to run the program and confirm whether your prediction was correct or not. If not try to explain which part of the code has behaved differently from what you predicted.

2.13 Modify the program to prompt for a 'c' to Cross, add a 'wait' conditional until 'c' key is pressed for the rest of the sequence. Adjust the timing and finally put a forever loop around the code to make it repeat.

LESSON 2: TRAFFIC SENSING TECHNOLOGY

ACTIVITY 2.1: SCRATCH PARTIAL CONTINUED



2.14 Your solution may look like:

```
when clicked
  forever
    switch costume to tl_Rcross
    wait 10 secs
    switch costume to tl_RA
    wait 5 secs
    switch costume to tl_G
    ask Press c to Cross? and wait
    wait until answer = c
    wait 1 secs
    switch costume to tl_A
    wait 8 secs
```

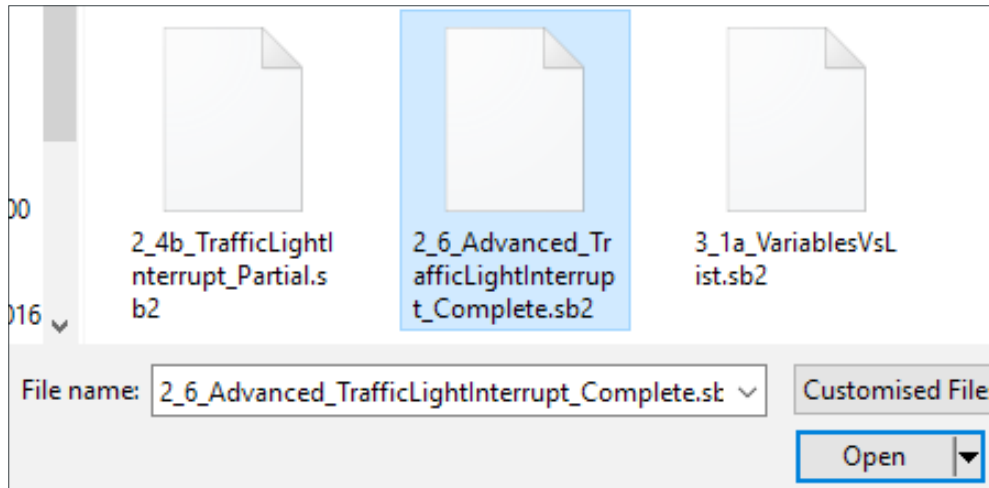
The image shows a Scratch script starting with a 'when clicked' event. It enters a 'forever' loop containing the following blocks: 'switch costume to tl_Rcross', 'wait 10 secs', 'switch costume to tl_RA', 'wait 5 secs', 'switch costume to tl_G', 'ask Press c to Cross? and wait', 'wait until answer = c', 'wait 1 secs', 'switch costume to tl_A', and 'wait 8 secs'. The loop ends with a return arrow block.

LESSON 2: TRAFFIC SENSING TECHNOLOGY

ACTIVITY 2.1: SCRATCH CHALLENGE

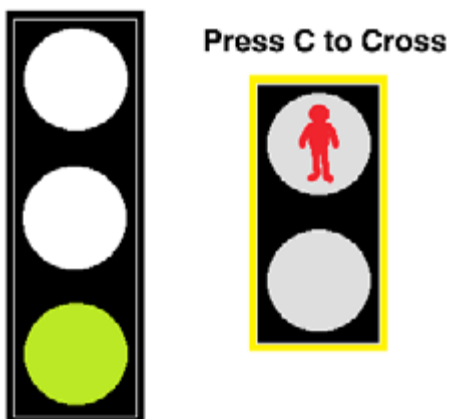


2.15 Load in the
2.6_Advanced_TrafficLightInterrupt_complete
program



2.16 Observe that this model has three sprites each with its own code. The traffic light for road uses, the pedestrian crossing and the message:

2_6_Advanced_TrafficLightInterrupt_Complete



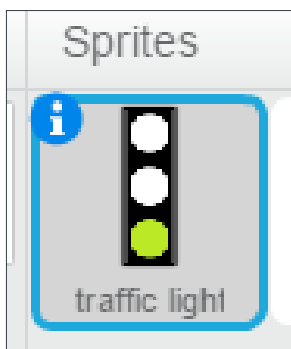
2.17 All of the code runs at the same time when the program is started by pressing the Scratch green flag, then the sprites pass messages (broadcasts) to each other to keep the sequence synchronised.

LESSON 2: TRAFFIC SENSING TECHNOLOGY



ACTIVITY 2.1: SCRATCH CHALLENGE CONTINUED

2.18 Run the program and follow which parts of the code are running before and after **you press the C key (on the Keyboard)** and how the messages are used to pass control of the program between the sprites. Adjust the timing to make it more realistic.



```

when clicked
  switch costume to tl_G

when I receive waiting to cross
  wait 2 secs
  switch costume to tl_A
  wait 2 secs
  switch costume to tl_R
  broadcast cross now
  wait 2 secs
  broadcast cross finished
  switch costume to tl_RA
  wait 2 secs
  switch costume to tl_G
    
```

```

when clicked
  switch costume to cr_red_man

when c key pressed
  broadcast waiting to cross

when I receive cross now
  switch costume to cr_green_man

when I receive cross finished
  switch costume to cr_red_man
    
```

```

when clicked
  switch costume to waiting

when I receive cross now
  switch costume to crossing

when I receive cross finished
  switch costume to waiting
    
```

LESSON 2: TRAFFIC SENSING TECHNOLOGY

RESOURCE 2.1: PEDESTRIAN CROSSING UNPLUGGED (WITH INTERRUPTION)



Objective

To decide on the traffic light sequence and timing so that pedestrians can cross and cars can stop safely after the crossing button is pressed.

In your group one student will hold the traffic light cards and act as the traffic light. Another student will hold the green man and red man cards and act as the pedestrian light. Another student should use the stopwatch or other timer to count the number of seconds on each phase of the light. A few other students should act as pedestrians and cars.

This is a more realistic simulation of a pedestrian crossing. The pedestrian crossing now DOES have a button to stop the traffic. The traffic lights will normally be continuously on green and only go into the sequence to stop the cars and allow pedestrians to cross after the button is pressed.

Try this out a few times and decide what time you need for each phase of the traffic lights in sequence

Red
Red/Amber
Green
Amber

Note this on the traffic light cards. Also note when the Green man and Red man cards should be shown.

Now simulate some different situations. For example

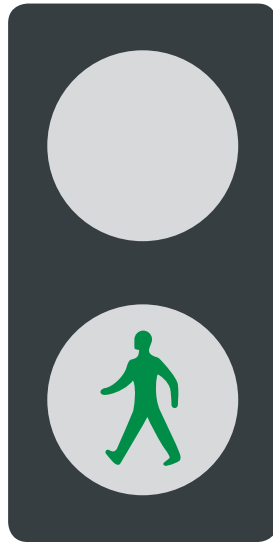
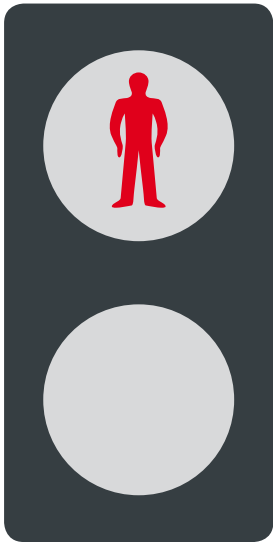
- ◆ many pedestrians crossing directly after each other
- ◆ long gaps with no pedestrians
- ◆ a slow crossing e.g. an elderly person

Think about how well your traffic light timing works in the different cases. Compare this to the fixed timing example in the previous lesson.



LESSON 2: TRAFFIC SENSING TECHNOLOGY

RESOURCE 2.2: GREEN MAN/RED MAN CROSSING CARDS



NOTES:

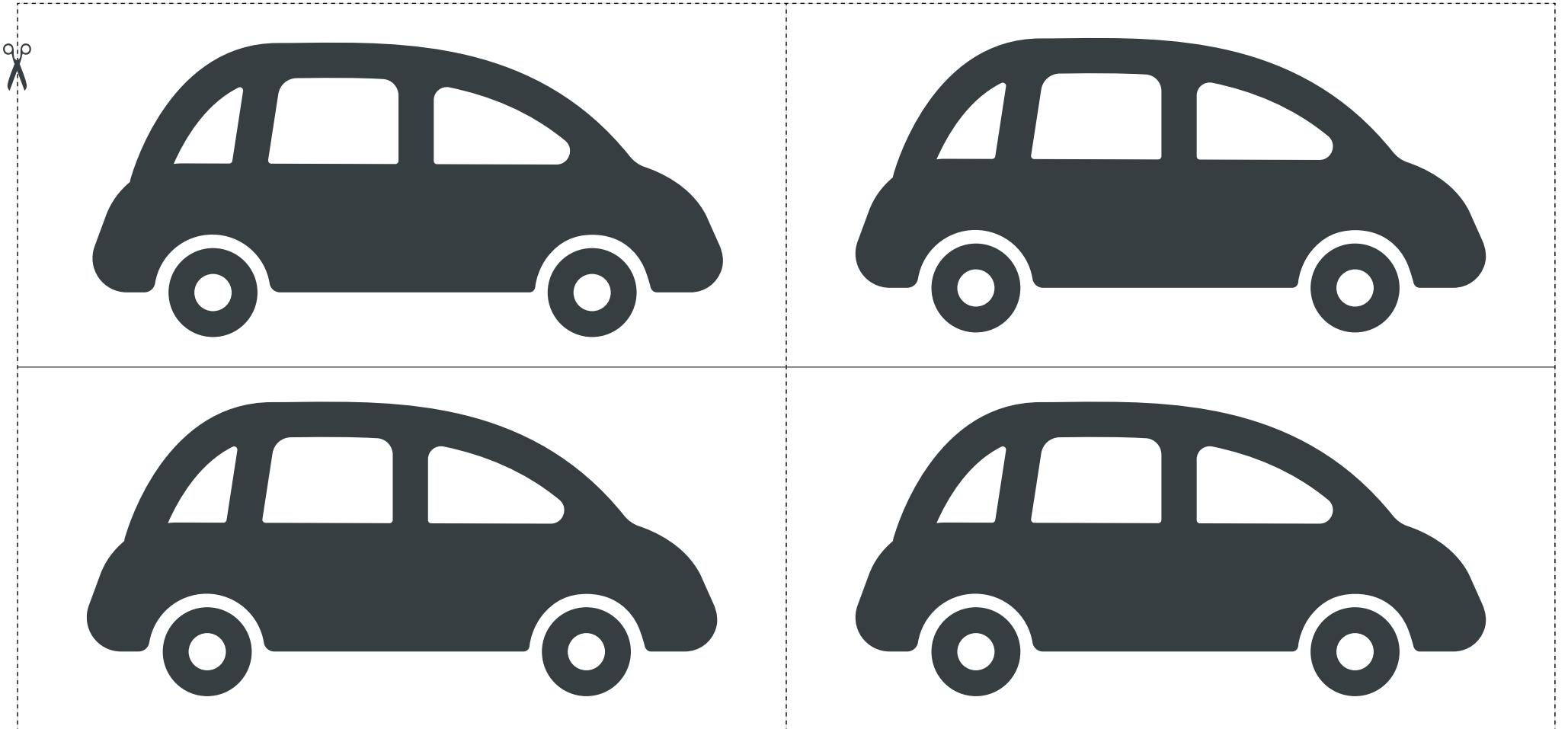
NOTES:

NOTES:

NOTES:

LESSON 2: TRAFFIC SENSING TECHNOLOGY

RESOURCE 2.3: CAR AND TRAFFIC LIGHT CARDS

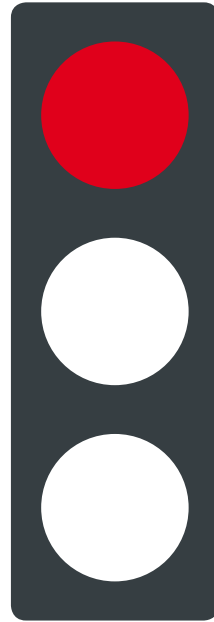


LESSON 2: TRAFFIC SENSING TECHNOLOGY

RESOURCE 2.3: CAR AND TRAFFIC LIGHT CARDS CONTINUED



Traffic light cards



TIMING:
NOTES:

TIMING:
NOTES:

TIMING:
NOTES:

TIMING:
NOTES:

LESSON 2: TRAFFIC SENSING TECHNOLOGY

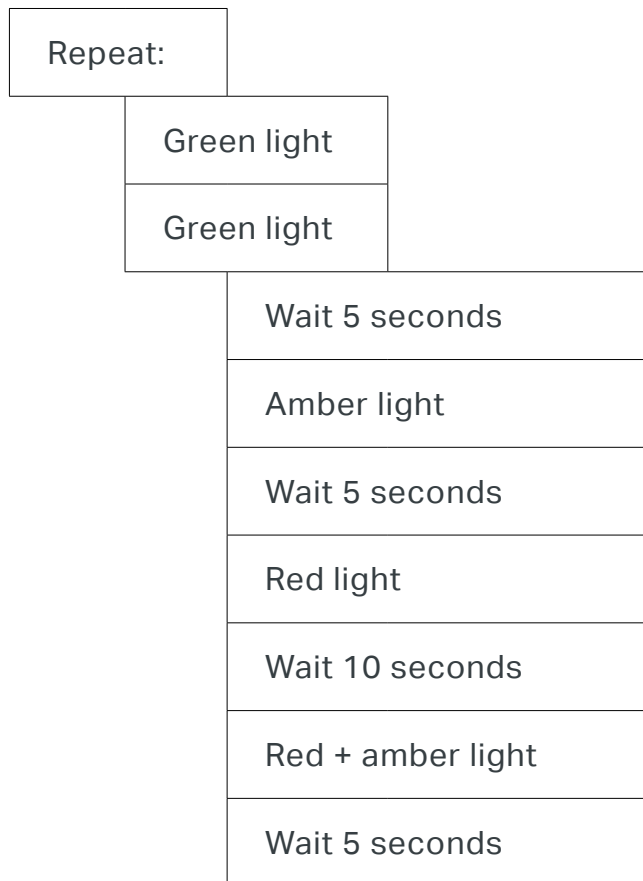
RESOURCE 2.4: ALGORITHM CARDS



Teacher instructions

Cut out the cards on the second page and ask students in groups to sort these out to represent the traffic light operation algorithm.

A sample sequence and timings are suggested here. Students should produce their own timings and justify that they provide a safe crossing for pedestrians.

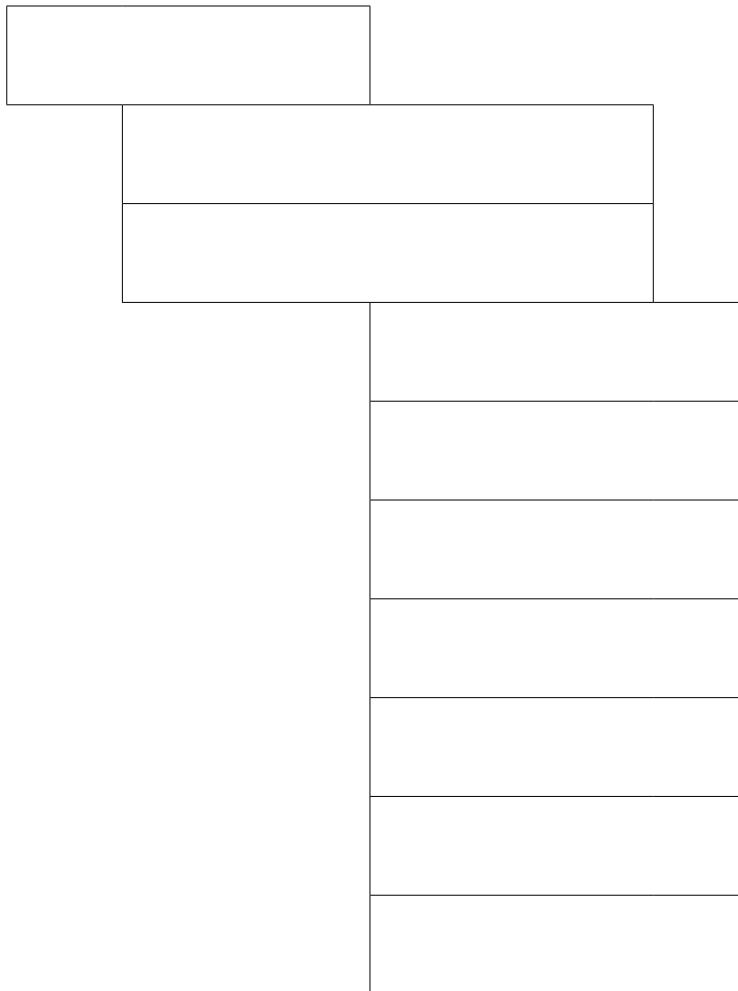


LESSON 2: TRAFFIC SENSING TECHNOLOGY

RESOURCE 2.4: ALGORITHM CARDS CONTINUED



Blank framework:



 **Repeat**

IF Crossing Button Pressed THEN

Light colour(s)

Light colour(s)

Light colour(s)

Light colour(s)

Light colour(s)


Light colour(s)

WAIT _____ seconds

WAIT _____ seconds

WAIT _____ seconds

WAIT _____ seconds

 **Repeat**

IF Crossing Button Pressed THEN

Light colour(s)

Light colour(s)

Light colour(s)

Light colour(s)

Light colour(s)

Light colour(s)

WAIT _____ seconds

WAIT _____ seconds

WAIT _____ seconds

WAIT _____ seconds

LESSON 3

WHERE'S THE BUS?



THE BIG IDEA

Transport for London, Facebook, Amazon, your favourite supermarket – they all handle vast amounts of data. Scratch enables us to make one or even several variables but it can't handle the sheer quantity of data that most modern organisations use. For that we need a more sophisticated program. A good way to start is with Python.



LEARNING OUTCOMES

Students:

Could describe what is meant by a key value pair in a dictionary

Could find the value for a key in a dictionary

Should use the type function to identify string and integer objects

Must create a list and add items to it. Recognise a dictionary.



RESOURCES

Students will need Python 3 installed on their computers.

It will be helpful, but not essential, for them to have a text editor, for example, *Notepad ++* for Windows or *Atom* for MAC OS.

Students will need two text files:

onebus.txt
more.txt

KEYWORDS

- ◆ Object
- ◆ String
- ◆ Data structure:
 - List
 - Dictionary

LESSON 3

WHERE'S THE BUS?



INTRODUCING PYTHON

We're going to be using **Python version 3**. This is the most up to date version and the one for which developers are producing new updates. Some scientists still use Python 2 for specialist work but we are going to use Python 3.

Python was devised by a person called Guido van Rossum. It seems he was a great fan of a comedy series called Monty Python's Flying Circus which has lots of jokes about spam. Not unwanted email but tinned meat.

Did you know?

Python is a "high level language". "High level" doesn't mean it is difficult, it means that it uses words that are similar to every day English.

LESSON 3

WHERE'S THE BUS?

SETTING THE SCENE

This lesson begins with some Python basics. Your students may be familiar with this material, in which case you can go through it fairly quickly. If not, students may benefit from spending several periods working on it.

The lesson then puts into practice handling data which has been downloaded from TfL's servers. Lesson 4 (page 73) will cover the downloading process but for this lesson students will work on data downloaded and prepared for them.



LESSON 3: WHERE'S THE BUS?

STUDENT PYTHON ACTIVITIES



MAIN 1

LOOK AROUND YOU – OBJECTS EVERYWHERE

Look around you in school or at home. There are objects everywhere: tables, chairs, computers, books, bags, even people.

It's the same in Python. Just remember that everything is an object. As you learn Python, we'll keep coming back to that.

We'll start our exploration of objects in Python with strings and integers. A string is one or more characters – letters, numbers, punctuation marks and special characters (such as \$ or \). An integer is a whole number, for example, 9.

To get started, open IDLE. IDLE is not the Python program, it is a program where you can try out snippets of code and get error messages as you learn. Technically, it is called an Integrated Development Environment.

```
Python 3.6.1 Shell
File Edit Shell Debug Options Window Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 18:41:36) [MSC
on win32
Type "copyright", "credits" or "license()" for more inform
>>> |
```

The cursor will be flashing, waiting for your input.

LESSON 3: WHERE'S THE BUS?

STUDENT PYTHON ACTIVITIES

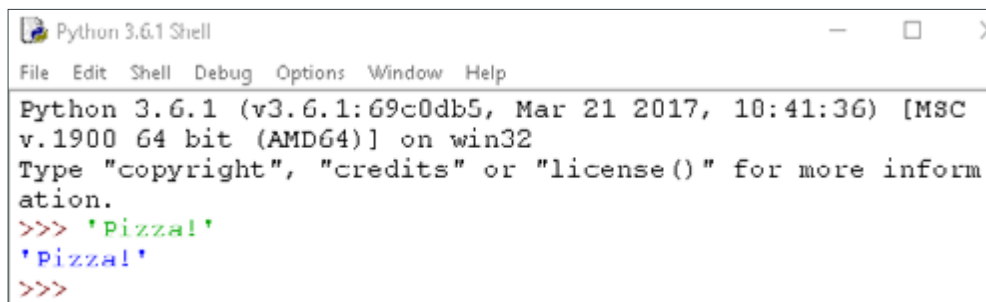


MAIN 1

LOOK AROUND YOU – OBJECTS EVERYWHERE CONTINUED

Python programmers often use food in their code. Spam is pink with lumps of fat, gross! Let's choose something else. Pizza!

The word Pizza! is a string – some letters and a punctuation mark. We have to enclose a string in single or double quotes.



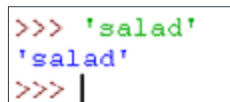
```
Python 3.6.1 Shell
File Edit Shell Debug Options Window Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 10:41:36) [MSC
v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more inform
ation.
>>> 'Pizza!'
'Pizza!'
>>>
```

After we enter 'Pizza!' IDLE repeats it back to us. What's happened?

Entering some characters (letters or numbers) kicks off a process called REPL – read, evaluate, print, loop. Python reads what we have input – Pizza! and evaluates it: it's a string. Python then prints it to the screen – in blue, with single quotes around it. Python then loops back, ready for whatever we enter at the prompt (the three arrows).

By the time the prompt appears, Python's garbage collector has "thrown away" our input.

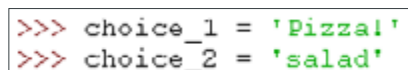
Let's try another string. Pizza is great but sometimes we want something a bit healthier. Let's try salad.



```
>>> 'salad'
'salad'
>>> |
```

It's the same process: REPL. Python reads our input – 'salad', evaluates it as a string, prints it on the screen - 'salad' – and loops back ready for whatever we input next. By the time the prompt appears, the garbage collector has swept up this string and thrown it away to free up memory.

If we don't want our input to be thrown away, we need to give each string a name.



```
>>> choice_1 = 'Pizza!'
>>> choice_2 = 'salad'
```


LESSON 3: WHERE'S THE BUS?

STUDENT PYTHON ACTIVITIES



MAIN 1

LOOK AROUND YOU – OBJECTS EVERYWHERE CONTINUED

Now we have two string objects – Pizza! and salad.

We assign 'Pizza!' to choice_1. choice_1 is a string object with the value Pizza!.

Similarly, choice_2 is a string object with the value salad.

We can see that choice_1 is a string object by entering

```
>>> type(choice_1)
```

When we press the Enter key Python responds:

```
<class 'str'>
```

Our object is a string.

Try it with a whole number.

```
>>> fave_number = 7
>>> type(fave_number)
<class 'int'>
```

But aren't these variables?

You may be familiar with a C based language (such as *Arduino*, *Java*, *Small BASIC*) where you have to declare (create) a variable and assign it a type before you can give it a value. If you declare a variable as an integer type, you can use it to store an integer but if you try to use it to store a string or a float (decimal number), you will generate an error.

In these languages you create a variable, decide what type it is and then use it.

A variable is a container. Once created, it can't be changed to a different type of container and it will only hold the values for which it was created.

Think about buying a new battery online for your phone. The firm that sells it will probably send it to you in a small padded bag because that is the most suitable package for a small item. If you order a new phone, it will come in a box because that is the most suitable package for that item. Variables are similar: you have to decide what to store in them.

Python is different. First you create the object. Then you give it a name.

A name is just a reference to an object.

LESSON 3: WHERE'S THE BUS?

STUDENT PYTHON ACTIVITIES



MAIN 1

LOOK AROUND YOU – OBJECTS EVERYWHERE CONTINUED

But aren't these variables? continued

You can have several names as references for the same object. Think about a relative. Maybe you have an aunt: you might call her "Aunty", other people may call her by her first name and other people might call her by a more formal name, e.g. Ms Brown.

This works in Python:

```
>>>  
>>> x = y = z = 9  
>>>
```

9 is an integer object.
x and y and z are all names for the same object.

But we could use x as the name for some food.

```
>>> x = 'fish fingers'  
>>> x  
'fish fingers'
```

x is now a name for the string object 'fish fingers'.

You can't use an integer variable to store a string in C based languages such as Arduino.

Bottom line: Python uses names for objects.

You will often see these called variables but they are not the same as variables in other languages. In particular, they are not containers. You can learn Python without using the term variables. If you want to use the term "variables", just remember that they are not the same as in other languages.

LESSON 3: WHERE'S THE BUS?

STUDENT PYTHON ACTIVITIES



MAIN 2

PYTHON – LISTS OF FOOD

We used `choice_1` and `choice_2` as names for two objects. Making a name for each food item would soon become boring and time consuming. Fortunately, Python has data structures that will help.

We could use a list. A list is an object which will hold other objects.

We use square brackets to create a list.

```
>>> []  
[]
```

This is not much use without a name. We have to choose a name, perhaps `main_courses`.

```
>>> main_courses = []  
>>>
```

We now have an empty list called `main_courses`.

Types and Classes

If you need convincing that this is a list, try entering

```
>>> type(main_courses)  
<class 'list'>
```

This tells us that `main_courses` is a list.

We can add food items to it, one at a time, using the `append` method.

```
>>> main_courses.append('Pizza')  
>>> main_courses.append('salad')  
>>> main_courses.append('fish and chips')
```

We can then use the `print` function to print to screen all of the items in the list.

```
>>> print(main_courses)  
['Pizza', 'salad', 'fish and chips']
```

You'll learn more – much more – about lists later.

LESSON 3: WHERE'S THE BUS?



STUDENT PYTHON ACTIVITIES

MAIN 3

PYTHON – FOOD AND PRICES IN A DICTIONARY

The school chef could use a Python list for all of the items on sale in the school canteen.

If she or he wants to record a price for each item, a Python dictionary might be more useful. It holds a key – value pair, for example, chips: £0.60. The food's name is the key, the price is the value.

We can create a blank dictionary by using a pair of curly braces and giving it a name.

```
>>> menu_prices = {}
```

It would probably be better to create a dictionary with some items in it.

```
>>> menu_prices = {'chips': 0.60, 'fish fingers': 0.75, 'baked potato': 0.35}
```

You can see the items in the dictionary by entering its name or by using the print function.

```
>>> print(menu_prices)
{'chips': 0.6, 'fish fingers': 0.75, 'baked potato': 0.35}
```

To add another item:

```
>>> menu_prices['burger'] = 1.20
```

You can check that this item is in the dictionary.

```
>>> print(menu_prices)
{'chips': 0.6, 'fish fingers': 0.75, 'baked potato': 0.35, 'burger': 1.2}
```

If you try this, the items may be displayed in a different order. If you have Python version 3.6.0 the order will stay the same.

LESSON 3: WHERE'S THE BUS?

STUDENT PYTHON ACTIVITIES



MAIN 3

PYTHON – FOOD AND PRICES IN A DICTIONARY

A dictionary is a very good data structure to use if you want to find an item and its value very quickly.

You can't really see how fast a dictionary is if you have only four items. But here's how to find the price of a burger.

```
>>> print(menu_prices['burger'])  
1.2
```

It's "instant" with four items. It's pretty much the same with thousands or even millions. A dictionary would be a good structure for storing all the mobile numbers from one operator (EE, Vodafone, O2, Three and so on).

You can't really see how fast a dictionary is if you have only four items. But here's how to find the price of a burger.

Challenge

Make a Python list for dessert.

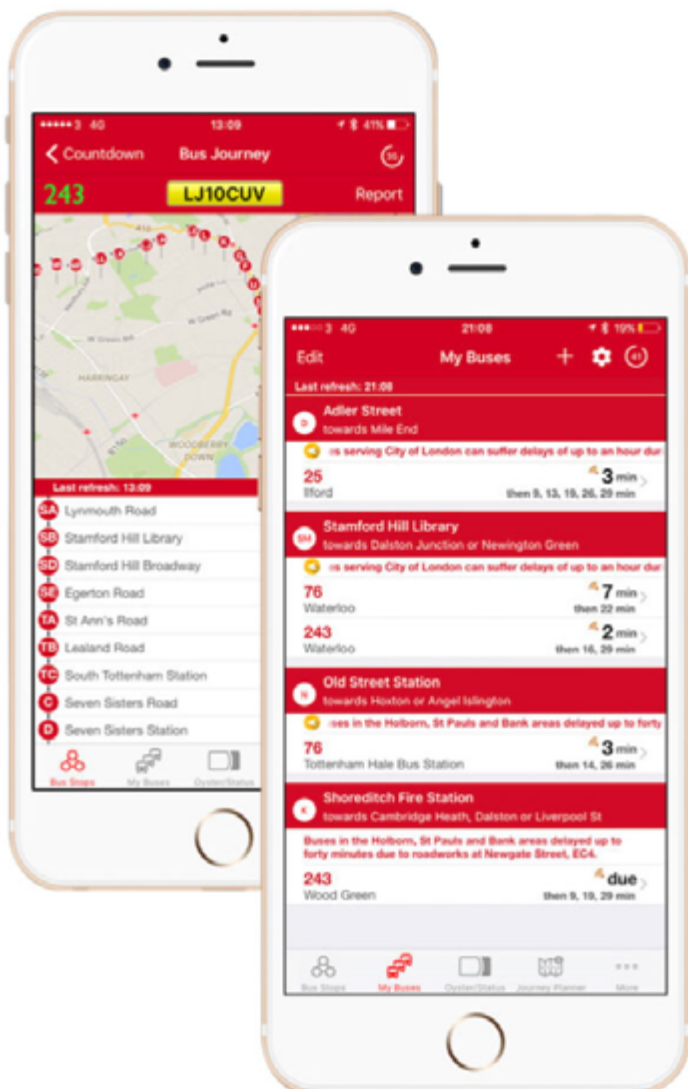
LESSON 3: WHERE'S THE BUS? STUDENT PYTHON ACTIVITIES



MAIN 4 INFO ON THE BUSES

After that quick trip into objects, lists and dictionaries, it's time to turn to the buses and real life journeys.

We all know that Transport for London has a large fleet of buses. To run an efficient service, TfL needs to be able to check on the progress of each bus at any time. Many bus stops have information displays, but passengers waiting at a stop that does not have a display can check arrival times on their phones. They can do this because TfL makes non confidential data available to developers who then make apps for mobiles.



LESSON 3: WHERE'S THE BUS?

STUDENT PYTHON ACTIVITIES

MAIN 4
INFO ON THE BUSES CONTINUED

Below is a screenshot of data on one red bus in London.

```
[
  {
    "$type": "Tfl.Api.Presentation.Entities.Prediction, Tfl.Api.Presentation.Entities",
    "id": "-1966001890",
    "operationType": 1,
    "vehicleId": "LJ13FBY",
    "naptanId": "490006728S",
    "stationName": "Falkland Road",
    "lineId": "29",
    "lineName": "29",
    "platformName": "H",
    "direction": "outbound",
    "bearing": "163",
    "destinationNaptanId": "",
    "destinationName": "Trafalgar Square",
    "timestamp": "2017-05-01T07:45:13Z",
    "timeToStation": 31,
    "currentLocation": "",
    "towards": "Manor House",
    "expectedArrival": "2017-05-01T07:45:44Z",
    "timeToLive": "2017-05-01T07:46:14Z",
    "modeName": "bus",
    "timing": {
      "$type": "Tfl.Api.Presentation.Entities.PredictionTiming, Tfl.Api.Presentation.Entities",
      "countdownServerAdjustment": "00:00:00.0060983",
      "source": "2017-04-29T04:43:33.683Z",
      "insert": "2017-05-01T07:44:08.855Z",
      "read": "2017-05-01T07:44:08.855Z",
      "sent": "2017-05-01T07:45:13Z",
      "received": "0001-01-01T00:00:00Z"
    }
  }
],
```

TfL presents its data in JSON format. As you can see, the data looks like a list with several objects - Python dictionaries. Do you remember that everything in Python is an object? A dictionary is an object. Here we have a list which contains several objects. Actually the list is also an object!

Keep hold of that idea: everything in Python is an object.

LESSON 3: WHERE'S THE BUS?



STUDENT PYTHON ACTIVITIES

If you are beginning to learn Python, it would be a bit much to expect you to work with raw data from TfL. You will find a simplified data set in a text file called `onebus.txt`. Open this file in a text editor, like Notepad or Notepad ++ and copy the data. You could use Ctrl-A to select all the data, then Ctrl-C to copy it.

Now open IDLE and paste in all the data.

```
>>> {
  "$type": "Tfl.Api.Entities.Transient.Prediction, Tfl.Api.Entities",
  "id": "57dc2d0a1e875a1638931f4e",
  "operationType": 1,
  "vehicleId": "LJ13FDN",
  "naptanId": "490004695D",
  "stationName": "Cambridge Circus",
  "lineId": "29",
  "lineName": "29",
  "platformName": "D",
  "direction": "outbound",
  "bearing": "159",
  "visitNumber": "1",
  "tripId": "3188",
  "vehicleCDId": "35451",
  "destinationNaptanId": "",
  "destinationName": "Trafalgar Square",
  "timestamp": "2016-09-16T17:41:00Z",
  "timeToStation": 1342,
  "currentLocation": "",
  "towards": "Parliament Square or Waterloo",
  "expectedArrival": "2016-09-16T18:03:22.5212712",
  "timeToLive": "2016-09-16T18:03:52.5212712",
  "modeName": "bus",
  "serverTimestamp": "2016-09-16T17:34:02.05Z"
}
```

Click before the first curly brace and enter a name for your dictionary, followed by an equals sign.

```
>>> one_bus = {
  "$type": "Tfl.Api.Entities.Transient.Prediction, Tfl.Api.Entities",
  "id": "57dc2d0a1e875a1638931f4e",
  "operationType": 1,
  "vehicleId": "LJ13FDN",
  "naptanId": "490004695D",
  "stationName": "Cambridge Circus",
```

Press the **Enter** key. You now have a dictionary. Not convinced? Enter

```
>>> type(one_bus)
<class 'dict'>
```

and press **Enter** to confirm that you have a dictionary object

LESSON 3: WHERE'S THE BUS?

STUDENT PYTHON ACTIVITIES



MAIN 5

ASKING QUESTIONS

You can now ask questions, using IDLE.

What's the registration number of this bus? The line

```
>>> one_bus['vehicleId']
```

returns a string. (Be sure to use square brackets.)

```
>>> one_bus['vehicleId']  
'LJ13FDN'
```

If you'd prefer not to have the quotes, use the print function:

```
>>> print(one_bus['vehicleId'])  
LJ13FDN
```

The print function displays results on screen, it does not send data to a printer.

Which route is the bus serving?

```
>>> print(one_bus['lineName'])  
29
```

Over to you to find more information about the bus.

Passengers on the 29 bus will know that sometimes it doesn't go all the way to Trafalgar Square. It can be annoying to find that it terminates at Mornington Crescent when you want to go further.

To find out whether the bus is going all the way to Trafalgar Square, enter

```
>>> one_bus['destinationName'] == 'Trafalgar Square'
```

Be very careful to use a double equals sign. A single equals sign gives a name to an object. A double equals sign is what we normally think of as equals.

Our bus is going all the way to Trafalgar Square:

```
>>> one_bus['destinationName'] == 'Trafalgar Square'  
True
```

Bus drivers have a really tough job and some are real heroes. We might like to find the name of the driver so that we can tell TFL how helpful he or she has been. Is driver information one of the items in the dictionary?

Just enter

```
>>> 'driver' in one_bus.items()
```

and press the **Enter** key to see **True** or **False**.

Seems that driver information is not part of the dataset.

LESSON 3: WHERE'S THE BUS?

STUDENT PYTHON ACTIVITIES



MAIN 6

BUSES, BUSES, BUSES – LISTS

Before you can find information from a list of dictionaries, you need to know – or revise – some basics on lists.

Here's a list of bus registration numbers, spotted at a junction in West London:

```
>>> bus_regs = ['YP59OEV', 'YR61RSO', 'LK55AAU', 'LK12AEA', \
                'LK12ACF', 'LK59CXC']
```

To see how many buses there are in the list, use the len() function.

```
>>> len(bus_regs)
6
```

There are six buses. OK, you could have just counted them because the list is very short but how easy would it be to count the number of buses that TfL has on the road each day? That's where methods come in.

Each item in a list – or each bus in our bus_regs list – has an index number. You can find an item by entering its index number in square brackets after the name of the list.

Try:

```
>>> bus_regs[1]
```

When you press Enter, you see the second bus – YR61RSO. That's because lists are zero indexed: the first item is at index zero.

LESSON 3: WHERE'S THE BUS?



STUDENT PYTHON ACTIVITIES

MAIN 6

BUSES, BUSES, BUSES – LISTS CONTINUED

Now try:

```
>>> bus_regs[0]
```

Do you see the first bus in the list?

`index()` is a Python method – just do a search on the internet for more information.

To display the bus registration numbers on screen, enter

```
>>> for each in bus_regs:  
    print (each)
```

Press Enter once to show that you have finished this for loop and press Enter again to run your code:

```
>>> for each in bus_regs:  
    print (each)  
  
YP590EV  
YR61RSO  
LK55AAU  
LK12ABA  
LK12ACF  
LK59CXC
```

You have written a loop which iterates through the list. “Iterates” means that it goes through the list, item by item, until it reaches the end.

LESSON 3: WHERE'S THE BUS?



STUDENT PYTHON ACTIVITIES

Have you ever stood at a bus stop and looked at the predicted times for all the buses that are expected to arrive over the next 20 minutes? TfL aims to keep track of where every bus is and when it is expected to arrive at each stop en route to its destination.



KILBURN PARK STATION BUS STOP
© Panhard 2010, Wikipedia Commons

Let's take a close look at all the data on one bus.

LESSON 3: WHERE'S THE BUS?

STUDENT PYTHON ACTIVITIES



This screenshot shows part of the data for bus YP590EV, which is serving passengers in West London. You can find the complete dataset in the file `more.txt`.

Here is a screenshot of info on two stops. OK, you can't read the detail but you can see that it's like a Python list that contains a number of objects – dictionaries.

You can open this text file – `more.txt` – using *Notepad* or *Notepad ++*. If you have a MAC you can use *Atom*, *TextWrangler* or our favourite, *TextMate*.

```
[
  {
    "$type": "Tfl.Api.Presentation.Entities.Prediction, Tfl.Api.Presentation.Entities",
    "id": "2110570219",
    "operationType": 1,
    "vehicleId": "YP590EV",
    "naptanId": "490010968V",
    "stationName": "Petersfield Road",
    "lineId": "e3",
    "lineName": "E3",
    "platformName": "V",
    "direction": "outbound",
    "bearing": "156",
    "destinationNaptanId": "",
    "destinationName": "Chiswick, Edensor Road",
    "LimeStamp": "2017-02-06T18:28:43Z",
    "timeToStation": 87,
    "currentLocation": "",
    "Towards": "Turnham Green",
    "expectedArrival": "2017-02-06T18:30:10Z",
    "timeToLive": "2017-02-06T18:30:40Z",
    "modelName": "bus",
    "timing": [
      {
        "$type": "Tfl.Api.Presentation.Entities.PredictionTiming, Tfl.Api.Presentation.Entities",
        "countdownServerAdjustment": "00:00:00.6801014",
        "source": "2017-02-06T14:39:41.717Z",
        "insert": "2017-02-06T18:27:55.492Z",
        "read": "2017-02-06T18:27:55.492Z",
        "sent": "2017-02-06T18:28:43Z",
        "received": "0001-01-01T00:00:00Z"
      }
    ]
  },
  {
    "$type": "Tfl.Api.Presentation.Entities.Prediction, Tfl.Api.Presentation.Entities",
    "id": "384873287",
    "operationType": 1,
    "vehicleId": "YP590EV",
    "naptanId": "4900012548",
    "stationName": "Wilkinson Way",
    "lineId": "e3",
    "lineName": "E3",
    "platformName": "A",
    "direction": "outbound",
    "bearing": "87",
    "destinationNaptanId": "",
    "destinationName": "Chiswick, Edensor Road",
    "LimeStamp": "2017-02-06T18:28:43Z",
    "timeToStation": 190,
    "currentLocation": "",
    "Towards": "Turnham Green",
    "expectedArrival": "2017-02-06T18:31:53Z",
```

LESSON 3: WHERE'S THE BUS?



STUDENT PYTHON ACTIVITIES

Open IDLE and enter a name for your list, followed by a space and an equals sign.

```
Type "copyright", "credits" or "license()" for more
>>> stops =
```

My list has the name '**stops**'

Then paste in the contents of the text file '**more.txt**'

```
File Edit Shell Debug Options Window Help
{id": "725755178",
"operationType": 1,
"vehicleId": "YP590EV",
"naptanId": "490004891E",
"stationName": "Cavendish School",
"lineId": "e3",
"lineName": "E3",
"platformName": "V",
"direction": "outbound",
"bearing": "71",
"destinationNaptanId": "",
"destinationName": "Chiswick, Edensor Road",
"timestamp": "2017-02-06T18:28:43Z",
"timeToStation": 1620,
"currentLocation": "",
"towards": "Edensor Road",
"expectedArrival": "2017-02-06T18:55:43Z",
"timeToLive": "2017-02-06T18:56:13Z",
"modeName": "bus",
"timing": {
  "$type": "Tfl.Api.Presentation.Entities.PredictionTiming, Tfl.Api.Presentation.Entities",
  "countdownServerAdjustment": "-00:00:00.6964463",
  "source": "2017-02-06T14:39:41.717Z",
  "insert": "2017-02-06T18:27:46.529Z",
  "road": "2017-02-06T18:27:46.529Z",
  "sent": "2017-02-06T18:28:43Z",
  "received": "0001-01-01T00:00:00Z"
}
}
```

Click at the end of the list (circled in the screenshot) and press **Enter**. You should see the three arrows, showing that IDLE is ready for you to enter some code.

LESSON 3: WHERE'S THE BUS?

STUDENT PYTHON ACTIVITIES



First, let's find out the number of stops for which TfL has predicted times. Do you remember the len() function?

```
>>> len(stops)
18
```

To find the names of the stops, we need to iterate through the list, using a for loop.

Here's a start:

```
>>> for each in stops:
```

Temporary name

each is just a name you choose for each item in a list. You choose the name. This would work just as well:

```
>>> for stop in stops:
```

We now need the print function to display info on screen. **We need to use the key stationName to get the value** which is the name of the bus stop.

Here's the complete code:

```
>>> for each in stops:
    print(each['stationName'])
```

Don't forget to press Enter to show that you have finished the for loop and then press Enter again to run the code.

LESSON 3: WHERE'S THE BUS?

STUDENT PYTHON ACTIVITIES



When you are on a bus you may want to know how long the journey is going to take.

We can develop our for loop to show the expected arrival time at each bus stop on our journey.

```
>>> for each in stops:
    print(each['stationName'] + ': ' + each['expectedArrival'])

Petersfield Road: 2017-02-06T18:30:10Z
Wilkinson Way: 2017-02-06T18:31:53Z
St Peter's Church: 2017-02-06T18:32:05Z
Rugby Road: 2017-02-06T18:34:17Z
```

Passengers don't need the time to fractions of a second. We can slice out the characters that make up the time to the hour and minute: the time in hours and minutes starts at character 11 and ends at character 16 – that's two characters for the hour, a colon, and two characters for the minute.

```
>>> for each in stops:
    print(each['stationName'] + ': ' + each['expectedArrival'][11:16])

Petersfield Road: 18:30
Wilkinson Way: 18:31
St Peter's Church: 18:32
Rugby Road: 18:34
```

That's better, but it looks a little ragged and untidy. If you have used placeholders with the print function, you could try:

```
>>> for each in stops:
    print('{:24} {:5}'.format(each['stationName'], each['expectedArrival'][11:16]))

Petersfield Road      18:30
Wilkinson Way        18:31
St Peter's Church    18:32
Rugby Road           18:34
```


LESSON 3: WHERE'S THE BUS?



STUDENT PYTHON ACTIVITIES

To finish this lesson, try to develop your print function to produce output like this:

```
The bus to Chiswick, Edensor Road is expected to arrive at Petersfield Road at 18:30
The bus to Chiswick, Edensor Road is expected to arrive at Wilkinson Way at 18:31
The bus to Chiswick, Edensor Road is expected to arrive at St Peter's Church at 18:32
The bus to Chiswick, Edensor Road is expected to arrive at Rugby Road at 18:34
The bus to Chiswick, Edensor Road is expected to arrive at Blandford Road at 18:35
```

Don't like this output? Challenge yourself with different formats to display predicted times for a bus route that you know.

LESSON 4

YOU'RE THE DEVELOPER

OVERVIEW

In Lesson 3 students learned or reviewed some Python basics and extracted information from a prepared file with data that originated from TfL servers.

In this lesson all students will be able to access the TfL API (Application Programming Interface) used by developers to extract information for the signs at bus stops and apps on mobiles, desktops and laptops that many of us use in our everyday lives.

The first section makes no demands on programming skills. Students will access data simply through a browser. Chrome is highly recommended.

The second section builds on programming skills from Lesson 3 (page 50).

Important

Students will need the Chrome browser to access the TfL API – or indeed other APIs.

While access through Internet Explorer or Edge is possible, we believe that Chrome will provide a better experience.

This lesson accesses data about hire bikes sponsored by Santander, see picture next page. As some students may not be aware of them the teacher should introduce them.

LESSON 4

YOU'RE THE DEVELOPER



THE BIG IDEA

Many of the bus stops in London have information displays showing the number of the next bus service and the number of minutes before it will arrive. These displays use data from TfL's servers, delivered over Internet. You may have an app on your phone to show when the next bus or train is due to arrive; phone apps use data from TfL's servers.

TfL is committed to providing data to developers where this is possible and legal. You can read more about this at tfl.gov.uk/info-for/open-data-users/our-open-data

Discussion

Throughout these lessons we encourage teachers and students to debate some of the implications of IT for our society.

Before starting to look at the data exposed by TfL through its API, ask students to take two minutes with a partner to discuss the type of data that TfL does not make publicly available and why.

Who could access data that is not publicly available and what limits do you think there would be on access?



BIKE DOCKING STATION

© Trevor Bragg

LESSON 4

YOU'RE THE DEVELOPER



LEARNING OUTCOMES

Students:

Could produce information from data using Python

Could retrieve data in JSON format

Should discuss what is meant by “up to date” in relation to the data available from TfL’s API

Must contribute to discussions on the relevance of their learning to everyday life

Must explore the TfL API using a browser.



RESOURCES

Students will need Python 3 installed on their computers.

They will need the Chrome browser (free and does not require admin access to install)

Teachers and students do not need to register with TfL to use the API.

LESSON 4: YOU'RE THE DEVELOPER

STUDENT ACTIVITIES



MAIN 1

EXPLORING TFL DATA

TfL makes data available through an API. API stands for Application Programming Interface. This is a set of commands that provides a way for developers to access data in their own programs.

Before accessing the API through your browser, work with a partner to make a list of the types of information you would expect TfL to provide. Bus times for certain, but what else? You can make your list on paper or in a word processing program – no need for Python!

Now go to <https://api.tfl.gov.uk> to see the range of data that developers can use. Although the Getting Started section at the top of the page says that developers should register for an Application ID and Key, you will be able to access many of the feeds on this page without registering.

How many of the types of data did you have on your list?

LESSON 4: YOU'RE THE DEVELOPER

STUDENT ACTIVITIES



MAIN 2

DATA ON THE CYCLE HIRE SCHEME

If you find that data on the cycle hire scheme is too slow to load, please skip to the next section on buses **Where's the Bus?** (page 85).

You will be learning how to access data on bikepoints in the Cycle Hire scheme. Click on **BikePoint**, then find **BikePoint** under **API REFERENCE**. Choose the first option: Gets all bike point locations. Choose **TRY**.

You will see **LOADING...**

Your computer is loading details of several hundred docking stations, so please be patient! It may take a couple of minutes.

LESSON 4: YOU'RE THE DEVELOPER

STUDENT ACTIVITIES



MAIN 2

DATA ON THE CYCLE HIRE SCHEME CONTINUED

You will see

Gets all bike point locations. The Place object has an additionalProperties array which contains the nbBikes, nbDocks and nbSpaces numbers which give the status of the BikePoint. A mismatch in these numbers i.e. nbDocks - (nbBikes + nbSpaces) != 0 indicates broken docks. GET

REQUEST URL

```
https://api-neon.tfl.gov.uk/BikePoint
```

RESPONSE BODY

```
[
  {
    "$type": "Tfl.Api.Presentation.Entities.Place, Tfl.Api.Presentation.Entities",
    "id": "BikePoints_1",
    "url": "https://api-neon.tfl.gov.uk/Place/BikePoints_1",
    "commonName": "River Street , Clerkenwell",
    "placeType": "BikePoint",
    "additionalProperties": [
      {
        "$type": "Tfl.Api.Presentation.Entities.AdditionalProperties, Tfl.Api.Presentation.Entities",
        "category": "Description",
        "key": "TerminalName",
        "sourceSystemKey": "BikePoints",
        "value": "001023",
        "modified": "2017-02-04T12:03:20.347Z"
      }
    ]
  }
],
```

You will see some general information which basically says that for each docking station, you will find the number of bikes, the number of docking positions and the number of spaces.

Next you will see the url – address on the internet – from which you have requested data by clicking GET.

The RESPONSE BODY has data on the docking stations; there are several hundred in London.

LESSON 4: YOU'RE THE DEVELOPER



STUDENT ACTIVITIES

Each BikePoint or docking station has an id. You can see the docking station with id BikePoints_1. It's ordinary, non-technical name is River Street, Clerkenwell.

Now scroll down to find how many bikes are available at the moment, how many docks are empty and the total number of docks at this docking station.

At 12 pm on 4 February, there were:

**14 bikes available, with 5 empty spaces; there were 19 docks –
14 with bikes, 5 empty.**

```
{
  "$type": "Tfl.Api.Presentation.Entities.AdditionalProperties, Tf
1.Api.Presentation.Entities",
  "category": "Description",
  "key": "NbBikes",
  "sourceSystemKey": "BikePoints",
  "value": "14",
  "modified": "2017-02-04T12:03:20.347Z"
},
{
  "$type": "Tfl.Api.Presentation.Entities.AdditionalProperties, Tf
1.Api.Presentation.Entities",
  "category": "Description",
  "key": "NbEmptyDocks",
  "sourceSystemKey": "BikePoints",
  "value": "5",
  "modified": "2017-02-04T12:03:20.347Z"
},
{
  "$type": "Tfl.Api.Presentation.Entities.AdditionalProperties, Tf
1.Api.Presentation.Entities",
  "category": "Description",
  "key": "NbDocks",
  "sourceSystemKey": "BikePoints",
  "value": "19",
  "modified": "2017-02-04T12:03:20.347Z"
}
```


LESSON 4: YOU'RE THE DEVELOPER

STUDENT ACTIVITIES



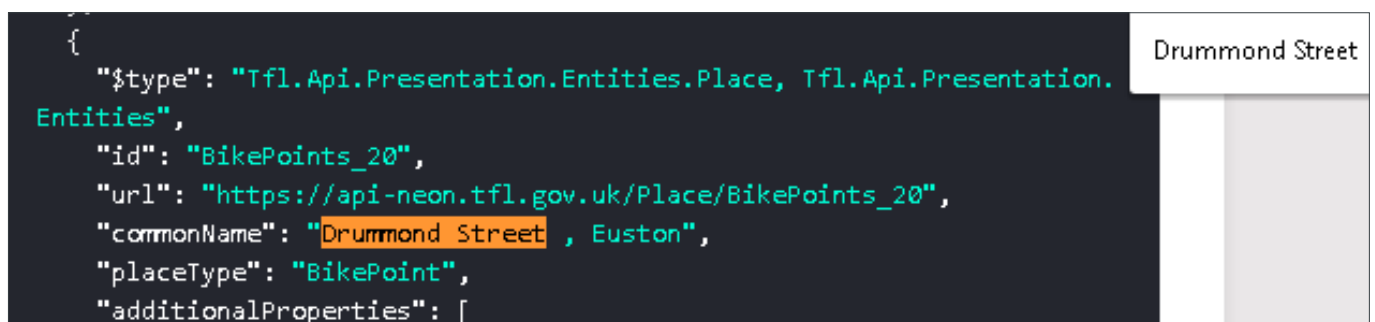
Each BikePoint has an id. Look carefully. Can you find 'id': **'BikePoints_2'**? Try using Ctrl-F in your browser and be sure not to enter any spaces.

As you scroll down the page, you will be able to see that each bikepoint has an id in the format **BikePoints_n** where n is a number.

Can you find where each bike point is by looking for its commonName key? Now choose a location where you know that there is a bikepoint and search for it using Ctrl-F in your browser. There is a bike point in Drummond Street.

```
{
  "$type": "Tfl.Api.Presentation.Entities.Place, Tfl.Api.Presentation.
Entities",
  "id": "BikePoints_20",
  "url": "https://api-neon.tfl.gov.uk/Place/BikePoints_20",
  "commonName": "Drummond Street , Euston",
  "placeType": "BikePoint",
  "additionalProperties": [

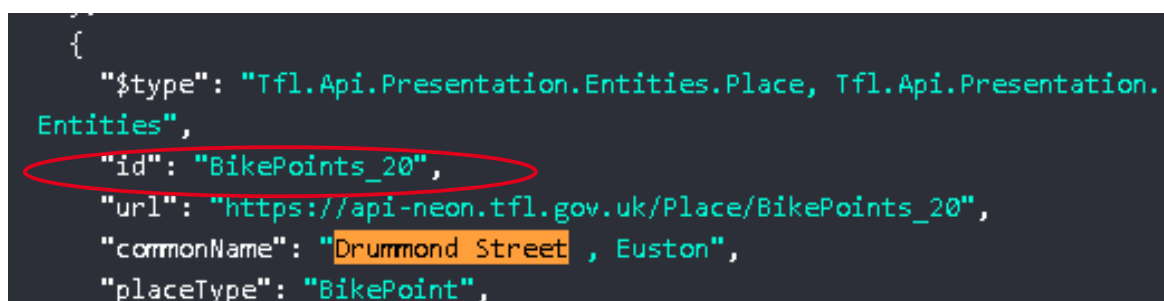
```



This screenshot shows that its id is BikePoints_20.

```
{
  "$type": "Tfl.Api.Presentation.Entities.Place, Tfl.Api.Presentation.
Entities",
  "id": "BikePoints_20",
  "url": "https://api-neon.tfl.gov.uk/Place/BikePoints_20",
  "commonName": "Drummond Street , Euston",
  "placeType": "BikePoint",

```



Just make sure that you remember the bikepoint id for the next search.

Close the browser tab to return to <https://api.tfl.gov.uk>

To find details of the bikepoint you found, choose BikePoint. A new page will open. Under API REFERENCE, click on BikePoint, then **Gets the bikepoint with the given id**.

LESSON 4: YOU'RE THE DEVELOPER

STUDENT ACTIVITIES



If you wanted information on the docking station with id: **BikePoints_30**, you would enter:

API REFERENCE

AccidentStats

AirQuality

BikePoint

Gets all bike point locations. The Pla...

Gets the bike point with the...

Search for bike stations by their nam...

id

Test this endpoint

TRY

Click on TRY. You will see:

Test this endpoint

LOADING...

Response Type application/json

The data will probably load really quickly. See if you can identify how many docking points there are at this location, how many bikes are currently available and how many points are empty.

LESSON 4: YOU'RE THE DEVELOPER

STUDENT ACTIVITIES



Here's the data for the docking station at Montgomery Square, Canary Wharf, which has the id **BikePoints_551**:

```
{
  "$type": "Tfl.Api.Presentation.Entities.AdditionalProperties, Tfl.
  Api.Presentation.Entities",
  "category": "Description",
  "key": "NbBikes",
  "sourceSystemKey": "BikePoints",
  "value": "16",
  "modified": "2017-05-03T20:43:14.853Z"
},
{
  "$type": "Tfl.Api.Presentation.Entities.AdditionalProperties, Tfl.
  Api.Presentation.Entities",
  "category": "Description",
  "key": "NbEmptyDocks",
  "sourceSystemKey": "BikePoints",
  "value": "23",
  "modified": "2017-05-03T20:43:14.853Z"
},
{
  "$type": "Tfl.Api.Presentation.Entities.AdditionalProperties, Tfl.
  Api.Presentation.Entities",
  "category": "Description",
  "key": "NbDocks",
  "sourceSystemKey": "BikePoints",
  "value": "39",
  "modified": "2017-05-03T20:43:14.853Z"
}
```

You can see the data in apps on mobiles and also in a browser.

LESSON 4: YOU'RE THE DEVELOPER

STUDENT ACTIVITIES



Here's how the data is displayed on a desktop PC using tfl.gov.uk/modes/cycling/santander-cycles/find-a-docking-station

The screenshot shows the TfL Santander Cycles website interface. At the top, there is a search bar containing the text "Montgomery Square, Canary Wharf" and a "Go" button. Below the search bar is a map of the area around Montgomery Square, Canary Wharf. A red pin is placed on the map at Montgomery Square, with a callout box that says "Montgomery Square, Canary Wharf (16 bikes)". Below the map is a bar chart showing the number of bikes available and spaces. The bar chart has 23 bars in total, with the first 16 bars colored red and the remaining 7 bars colored white. Below the bar chart, it says "16 bikes available" and "23 spaces".

Discussion point

Many of the apps you probably use were developed because someone, often working with a partner or group, wanted something extra or something better to solve a problem.

What's that song? Who wrote it? Welcome, the app *Shazam!*

Take two minutes to discuss with a partner: what additional information would you expect to find in an app on your phone or tablet?

LESSON 4: YOU'RE THE DEVELOPER

STUDENT ACTIVITIES



How Up to Date?

Data is only as good as the last update.

Can you find when the data on the bike points was last updated? You won't find the term "updated" but you will find a word that is relevant.

Now think about data on individual buses.

Take three minutes with a partner to discuss why TfL needs to know where each bus is. Is it true that you wait ages for a bus, then three come along at once? Should we really be annoyed when a bus stops for a minute or two "to regulate the service"? How often do you think the location of each bus should be refreshed?

In the next section you will be looking at data on buses. You will be able to find how often the location of each bus is updated.



LESSON 4: YOU'RE THE DEVELOPER

STUDENT ACTIVITIES



Where's the Bus?

In the Vehicle section, you can find where a bus is, which route it is serving and when it is expected to arrive at its destination.

Gets the predictions for a given list of vehicle Id's.

Show samples

GET /Vehicle/{ids}/Arrivals

Parameters

ids

array

A comma-separated list of vehicle ids e.g. LX58CFV,LX11AZB,LX58CFE. Max approx. 25 ids.

Test this endpoint

TRY

Response Type

LESSON 4: YOU'RE THE DEVELOPER



STUDENT ACTIVITIES

Here are a few bus registration numbers, spotted in West London:

YP59OEV, YR61RSO, LK55AAU, LK12AEA, AK12ACF, LK59CXC, BF10LTA.

Before the next lesson, note down a few bus numbers. When you enter bus numbers, make sure that you do not leave any spaces. You might get an error like this:

```
RESPONSE BODY
[]

RESPONSE CODE
200
```

Take two minutes to discuss with a partner possible reasons for this error message. You could get this message even when you are totally certain that you have entered a bus registration number correctly and had it checked by a partner.

Why might this be?

LESSON 4: YOU'RE THE DEVELOPER

STUDENT PYTHON ACTIVITIES



MAIN 3

ACCESSING DATA THROUGH PYTHON

Why am I learning this?

You may be thinking that someone has already built apps for bikes, buses and trains, so why am I learning all this?

Nothing stays the same. Just as we have all become familiar with the red dot matrix displays at most bus stops, there's a new type of display that does not require a mains electricity supply because it works from a battery (which could perhaps be solar powered). It shows TFL's buses and also buses from other companies. Take two minutes to discuss with a partner what other information you might like to see, especially if you are waiting a long time for a bus in bad weather.

Now take two minutes with another partner to make a list of apps and internet based services that have been developed in the last few years.

You might have identified Shazam, Twitter or Deliveroo. Twitter has offices in London with programmers, designers and researchers who travel to major events that the company reports. IT and computer science are always transforming the world we live in. What will come next and what sort of skills will be needed? Not just programmers.

Accessing data through python

You are now going to access data on the bikepoints using Python.

You will need to import a couple of modules into Python. You could think of modules as being like kits of extra commands. The modules you will be using are "built into" Python which is why sometimes you may see Python described as "coming with batteries". It's not like something you might buy or get as a present that you can't use until you have bought some batteries.

LESSON 4: YOU'RE THE DEVELOPER

STUDENT PYTHON ACTIVITIES



A quick look at modules

To start becoming familiar with modules, open IDLE which is an editor for Python and enter this code:

```
>>> import random
>>> die = random.randint(1, 6)
>>> print(die)
```

This simulates throwing a die. The line

```
>>> print(die)
```

will display a random number between 1 and 6.

Repeat the lines

```
>>> die = random.randint(1, 6)
>>> print(die)
```

to display another number at random.

You could display ten random numbers with this code:

```
>>> import random
>>> for i in range(10):
    die = random.randint(1, 6)
    print(die)
```

Press Enter twice – once to show that you have finished the for loop and once more to run the code.

Here are my numbers:

```
2
1
5
1
4
3
2
2
2
3
```

No sixes, must try again....

If you write any sort of games or quizzes, you will probably use the random module.

LESSON 4: YOU'RE THE DEVELOPER

STUDENT PYTHON ACTIVITIES



Modules to import data

The program to read data from the TfL API needs two modules:

1. to request data from the URL (web link)
2. to display it in JSON format.

JSON is a format for data that is easy to read for humans and which is logically organised for analysis in a computer program.

Start Python (version 3) and open **bikepoints_part1.py**.

```
File Edit Format Run Options Window Help
# bikepoints_part1.py

# import modules for data access
import urllib.request
import json

# data source
url = 'https://api.tfl.gov.uk/BikePoint'

# Get data for all bikepoints
response = urllib.request.urlopen(url)
bike_data = response.read()
```

You can see the two lines that import the modules you will need.

The data source has been given the name **url**. **url** is just a name: you could use any sensible name, such as **source**. If we changed the name in line 8, we would have to use the same name in line 12.

Line 12 uses the `urllib.request` module to open the URL (**api.tfl.gov.uk/Bikepoint**) and saves the data it retrieves with the name **response**.

The next line reads what has been saved with the name **response** and gives it the name **bike_data**.

Choose **Run > Run Module**. You should not see any results but if you see an error message, go back and check for typos.

bike_data is an object. You have used `type(object)` to find what type of data we have.

You may have used

```
>>> type(main_courses)
<class 'list'>
```

to confirm that you had made an empty list for food.

LESSON 4: YOU'RE THE DEVELOPER

STUDENT PYTHON ACTIVITIES



Modules to import data continued

In the program `bikepoints_part1.py`, add the line with the yellow outline:

```
# Get data for all bikepoints
response = urllib.request.urlopen(url)
bike_data = response.read()
print(type(bike_data))
```

Now run the program again.

Python has imported the data as bytes or sequences of numbers representing characters.

We need to load the bike data in JSON format, using UTF8 encoding which is standard for webpages.

```
bike_data = json.loads(bike_data.decode('utf8'))
print(type(bike_data))
```

Running the Python script, you will see that `bike_data` is now a list.

```
<class 'bytes'>
<class 'list'>
```

There are many list operations. One of these is `len()` which will tell us how many objects there are in the list.

```
print(len(bike_data))
```

Running the Python script, you will see the number of docking stations.

```
<class 'bytes'>
<class 'list'>
774
```

There may be more – or fewer – by the time that you run the script.

LESSON 4: YOU'RE THE DEVELOPER

STUDENT PYTHON ACTIVITIES



Modules to import data continued

In Lesson 3: Where's the bus? (page 50) you used a for each loop to display details of each bus.

The next section of code goes through the list, displaying for each docking station its id and its commonName (which is its location).

```
for each in bike_data:
    print('Station: {}, Location: {}'.format(each['id'], each['commonName']))
```

Here's the complete program tidied up a little:

```
# bikepoints_part1.py

# import modules for data access
import urllib.request
import json

# data source
url = 'https://api.tfl.gov.uk/BikePoint'

# Get data for all bikepoints
response = urllib.request.urlopen(url)
bike_data = response.read()
bike_data = json.loads(bike_data.decode('utf8'))

print('Number of docking stations: {}'.format(len(bike_data)))
input('Press Enter to continue.')

for each in bike_data:
    print('Station: {}, Location: {}'.format(each['id'], each['commonName']))
```

LESSON 4: YOU'RE THE DEVELOPER

STUDENT PYTHON ACTIVITIES



MAIN 4

IT'S ALL IN THE DETAIL

If you have reached this point, take a moment to congratulate yourself. You've probably tackled a lot of new material. That's important because the world is changing at an increasing rate and we all have to learn new ideas and skills in all areas of our lives.

In this activity you will get details of a docking station.

Open the Python script you made earlier or use **bikepoint_part2_start.py** to give yourself a fresh copy.

Add two blank print functions to create two blank lines after the list of docking stations has been displayed.

```
# bikepoints_part2 start.py

# import modules for data access
import urllib.request
import json

# data source
url = 'https://api.tfl.gov.uk/Bikepoint/'

# Get data for all bikepoints
response = urllib.request.urlopen(url)
bike_data = response.read()
bike_data = json.loads(bike_data.decode('utf8'))

print('Number of docking stations: {}'.format(len(bike_data)))
input('Press Enter to continue')

for each in bike_data:
    print('Station: {0}, Location: {1}'.format(each['id'], each['commonName']))

print()
print()
```

LESSON 4: YOU'RE THE DEVELOPER

STUDENT PYTHON ACTIVITIES



MAIN 4

IT'S ALL IN THE DETAIL CONTINUED

Now ask the user for a bike point id. The user's answer is saved in memory with name **bikepoint_id**.

```
bikepoint_id = input('Enter id for a docking station in format BikePoints_N: ')
```

You may remember that when you were accessing the API through your browser, you entered the bikepoint id in this format:

The screenshot shows an API reference page. On the left, under 'API REFERENCE', there is a list of endpoints: AccidentStats, AirQuality, BikePoint, and Search for bike stations by their nam... The 'BikePoint' endpoint is highlighted. Below it, the description reads: 'Gets the bike point with the...'. On the right, the 'id' parameter is shown with a value of 'BikePoints_30'. Below the input field, it says 'string' and 'A bike point id (a list of ids can be obtained from the above BikePoint call)'. At the bottom, there is a 'Test this endpoint' section with a blue 'TRY' button.

The next line of code requests data for an individual bikepoint using the url from the first part of the program with a forward slash and the bikepoint id added.

```
response_for_id = urllib.request.urlopen(url + '/' + bikepoint_id)
```

The next two lines are almost the same as those you wrote earlier to save the data for the bikepoint in JSON format with the name **station_data**.

```
station_data = response_for_id.read()  
station_data = json.loads(station_data.decode('utf8'))
```

LESSON 4: YOU'RE THE DEVELOPER

STUDENT PYTHON ACTIVITIES



MAIN 4 IT'S ALL IN THE DETAIL CONTINUED

You now need to display basic information about the docking station.

```
print('BikePoint ID:', station_data['id'])
print('Common name:', station_data['commonName'])
print('Latitude:', station_data['lat'])
print('Longitude:', station_data['lon'])
```

For each docking station there is additional information held in a dictionary called `additionalProperties` in key-value pairs.

This line of code simply displays the text

Additional information:

```
print('Additional information:')
```

The final three lines go through the dictionary **`additionalProperties`** to look for these three keys:

- NbBikes** the number of bikes available for hire
- NbEmptyDocks** the number of empty docks to which bikes can be returned
- NbDocks** the total number of docking points: this is the number of bikes that the docking station holds when full

If the program finds these keys, it prints them out with their values.

Here are the final three lines:

```
print('Additional information:')
for info in station_data['additionalProperties']:
    if info['key'] in ['NbBikes', 'NbEmptyDocks', 'NbDocks']:
        print('\t' + info['key'] + ': ' + info['value'])
```

`\t` adds a tab character to indent the key and its value.

And here is a sample output:

```
Additional information:
  NbBikes : 0
  NbEmptyDocks : 26
  NbDocks : 26
```

It's home time and all the bikes are out on hire.

You can find the completed script saved as **`bikepoints_part2_complete.py`**

LESSON 4: YOU'RE THE DEVELOPER

STUDENT PYTHON ACTIVITIES



EXTENSION ACTIVITY

You may have followed along and entered the code section by section, trying to understand it.

Or you may have jumped ahead and just downloaded the complete version.

Your challenge now is to write a program to display the route, current location and destination for a bus when the user enters its registration number. You may wish to display the registration number too, as confirmation to the user that the program has found the information for the correct bus. Data is of no value unless it is up to date: it would be a good idea to display the timestamp on the data – meaning when it was produced.

To get started, use the API to find the format of the URL and the format in which the data is returned. The rest is up to you.

LESSON 4: YOU'RE THE DEVELOPER

STUDENT PYTHON ACTIVITIES



EXTENSION ACTIVITY CONTINUED

First attempt

```
File Edit Format Run Options Window Help
# bus_info.py

# import modules for data access
import urllib.request
import json

# data source
url = 'https://api.tfl.gov.uk/Vehicle'

vehicleId = input('Please enter registration number with no spaces: ')
response_for_id = urllib.request.urlopen(url + '/' + vehicleId + '/Arrivals')

bus_data = response_for_id.read()
bus_data = json.loads(bus_data.decode('utf8'))

print('Registration number: {}'.format(bus_data[0]['vehicleId']))
print('Data provided at: {}'.format(bus_data[0]['timestamp'][11:16]))
print('Route: {}'.format(bus_data[0]['lineName']))
print('Now at: {}'.format(bus_data[0]['stationName']))
print('Destination: {}'.format(bus_data[0]['destinationName']))
```

LESSON 4: YOU'RE THE DEVELOPER

STUDENT PYTHON ACTIVITIES



EXTENSION ACTIVITY CONTINUED

Second attempt

The first attempt works provided that the user inputs a valid registration number. If the number is not valid for any reason, the program crashes - with lines in red if you are using IDLE.

Some students may have used try except blocks to deal with errors. Here's a version that handles invalid bus registration numbers.

```
File Edit Format Run Options Window Help
# bus_info_error_handling.py

# import modules for data access
import urllib.request
import json
import sys

# data source
url = 'https://api.tfl.gov.uk/Vehicle'

vehicleId = input('Please enter registration number with no spaces: ')
try:
    response_for_id = urllib.request.urlopen(url + '/' + vehicleId + '/Arrivals')
except urllib.error.HTTPError:
    print('')
    Please check the registration number. If it is correct, the bus may be
    out of service.
    ''')
    sys.exit()

bus_data = response_for_id.read()
bus_data = json.loads(bus_data.decode('utf8'))

print('Registration number: {}'.format(bus_data[0]['vehicleId']))
print('Data provided at: {}'.format(bus_data[0]['timestamp'][11:16]))
print('Route: {}'.format(bus_data[0]['lineName']))
print('Now at: {}'.format(bus_data[0]['stationName']))
print('Destination: {}'.format(bus_data[0]['destinationName']))
```

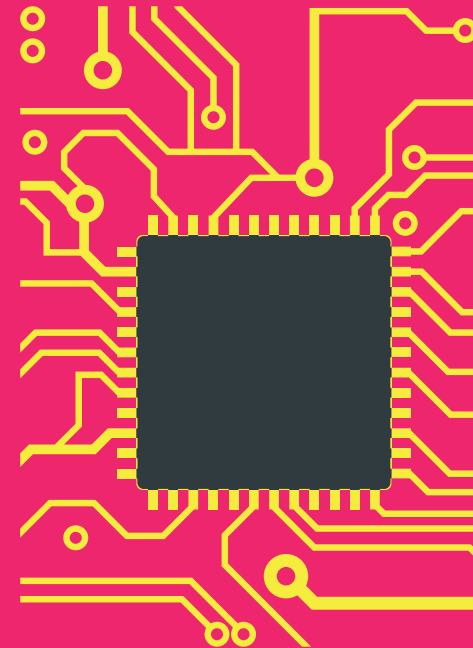
EXPLORE

Students will have worked with programs and data representing abstract or smaller parts of real systems. The Explore visits should show how the principles and techniques they learn within the curriculum are applied to real world problems and systems. They may see a range of technologies and programming languages used that they will not see in the classroom but should still see that engineers apply a computational thinking approach to problem solving in the first instance.

Transport for London have a number of locations relevant to road transport and traffic signals that schools can visit that will relate directly to topics in the Discover lessons and the overarching theme of computational thinking.

An alternative visit to the Crystal building Keep Moving exhibition will focus more on the ideas and outcomes of Discover lesson 4 of an integrated transport system rather than on computational problem solving. This visit may be undertaken with visits for other London Curriculum units to the Crystal building.

Alternatively other trips could be made where the use of technology has been designed to solve a problem through computation that students can either analyse themselves or have described or explained to them, particularly where an engineer can describe problems they are trying to solve and their approaches to solving them.



LESSON 5

EXPLORING TRANSPORT SYSTEMS



THE BIG IDEA

Analysing problems and finding solutions lie at the heart of computer science. These processes are vital for students whatever direction their future lives will take them.

A visit to the London Transport Traffic Control Centre or a major transport hub will enable students to see the relevance of their learning. They will also be able to evaluate the usefulness of the technology they see and to consider improvements and enhancements.



Planning before the visit

With the whole class, discuss / describe the range of methods for planning a journey in London, which might include the TfL journey planner online, mobile apps, phone call to TfL, enquiry at a station, asking a friend or family member.

Divide students into small groups to discuss how to reach the Explore location.

Discussion points

Which is more convenient – the TfL Journey planner online or an app?

How easy is it to customise the online planner or app for your preferences, for example, your preferred method of transport or accessibility requirements?

Taking as an example a journey with which students are familiar, does their

chosen planner produce the results they expect? If not, is the suggested route an improvement on their usual choice of route or not? Why?

Most people have limited data plans for their mobiles. Students will have seen the data used for queries about bus times or the availability of bikes. How does the data size for these queries compare with the size of a song or video? Will using a journey planner app consume much of a typical data allowance?

Are there other factors that the planner could take into account?

LESSON 5: EXPLORE TRANSPORT SYSTEMS



ACTIVITY

MAKE YOUR OWN WEBPAGE

Most students will have had at least some experience of authoring webpages. They can quickly and easily embed a widget from TfL into a webpage to use TfL open data.

Here are the steps to author a simple webpage to check the status of the tube lines:

1. First create a folder.
2. Using a simple text editor such as *Notepad ++*, create a file called **index.html**
3. Use a basic template



```
index.html x
1  <!doctype html>
2
3  <html lang="en">
4  <head>
5    <meta charset="utf-8">
6    <title>Your title here</title>
7  </head>
8
9  <body>
10
11 </body>
12
13 </html>
14
```

This file is available in the supplied resources; alternatively, students will be able to find basic templates online, for example, at www.sitepoint.com/a-basic-html5-template/

4. Change the text inside the <title> </title> tags to add the title that will appear in the browser tab. For example, enter Tube Updates.
5. Visit the TfL open data widget page at tfl.gov.uk/forms/12425.aspx
6. Copy the embed code for Tube service updates and paste it between the <body> </body> tags.
7. Save the file.
8. Double click the file to open it in a browser.

LESSON 5: EXPLORE TRANSPORT SYSTEMS

POSSIBLE SITES

A visit to a major station offers many opportunities to see technology in use, to assess which forms of technology are most useful to travellers and to consider enhancements and innovative solutions. Students could be organised into groups of 4–5 to carry out investigations, using suggested topics below.

It is important for students to realise that TfL does not own or manage the stations with interchanges to national rail, for example, Kings Cross St Pancras, London Bridge or Paddington.



KINGS CROSS STATION

© Transport for London



cactusbeetroot 2011, flickr



PADDINGTON STATION

givingnot@rocketmail.com 2011, flickr

LESSON 5: EXPLORE TRANSPORT SYSTEMS

POINTS FOR GROUPS TO INVESTIGATE

How easy is it to plan a journey at a station?

Visitors arriving from outside London will find posters with tube and rail lines. Students could watch – discreetly – to see how easy people find these to use. If your chosen station is Paddington, students could try the interactive map at the (relatively) new Hammersmith and City station at Paddington (platforms 15 and 16). You can read about this at

www.acquiredigital.com/blog/2016/5/16/tfls-new-interactive-map-experience

Is this genuinely useful technology or an expensive solution to a problem that does not really exist?

How easy is it to find out about problems on your route?

Arriving at a platform, travellers will see the expected time to wait for the next 3 trains. This is helpful but many journeys involve changing to another line. How easy is it to find whether there are any delays on other lines?

How easy is it to use a mobile device (phone or tablet)?

Many or most of us plan journeys on mobiles but how easy is it to use these at a station? What suggestions do students have for making mobile access easier? Are there downsides?

How much do we want to use contactless technologies? Most people who travel on the tubes, buses or TfL rail use some form of contactless technology, either an oyster card or a phone with NFC. Both use radio-frequency identification (RFID) technology.

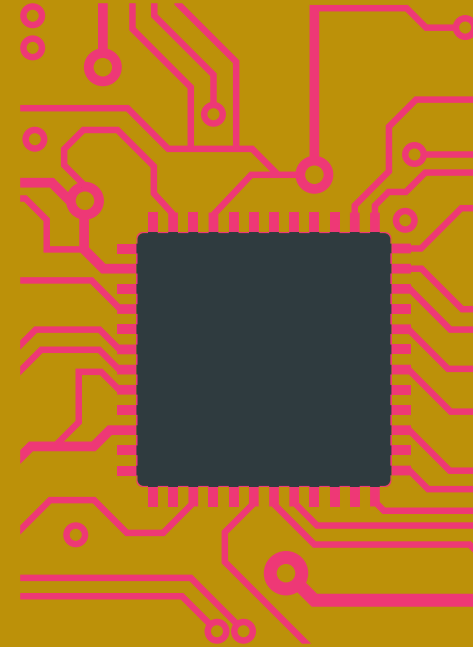
Most stations have cafés or stalls selling coffee. What proportion of customers use contactless cards/ phones compared with those who use cash or non-contactless cards?

Students may find advertising posters with QR codes or NFC tags. Do any travellers actually use these codes or tags with their phones? Why or why not?

CONNECT

Students will connect the computational problem solving they have worked on in the Discover lesson with a real London context either with Transport for London examples and data or with a local example or an engineer from a London organisation.

In each case students will identify the problem that needs solving and aspects of decomposing the problem and identifying the data that is needed to work towards a solution



LESSON 6

COMPUTING LONDON SOLUTIONS



BIG IDEA

Understanding how modern computer technology, computational thinking and programming are used to solve problems in London.



LEARNING OUTCOMES

Students:

Could independently suggest and evaluate problem decompositions and data requirements.

Should identify the benefits of the computing technology for the problem being addressed and understand the decomposition of the problem

Must produce a summary of the use of computing technology in an application or in the work of an engineer either previously covered or from new research



RESOURCES

Resource C1: Reporting framework example

LESSON 6: COMPUTING LONDON SOLUTIONS

SETTING THE SCENE

Computing London solutions

Students will have seen some of the advances in computing technology used in traffic monitoring and control and the work of engineers within London's transport systems, from alternative types of detector to open data all connected by networks. They should recognise that the changing nature and demands of London, including population and the promotion of public over private transport, have required innovative new solutions. The innovations that keep London moving are developed by engineers using computational thinking to analyse and solve new problem requirements.

This Connect lesson asks students to look at one situation and some of the computational thinking involved. The suggestions given are for a transport related or other engineering problems, an app using open data or the use of computing in the work of an engineer. Other examples may be chosen provided they bring out the computational thinking element.



Tadie88 2015, flickr

LESSON 6: COMPUTING LONDON SOLUTIONS

ACTIVITIES

STARTER

Setting the task

Groups of students in the class may work on one or more of the following tasks or a similar task of your choosing and present a final report.

1. Transport system or other engineering problem – research a problem, describe the main problem, decompose into the main modules needed for a solution. Describe the hardware, particularly sensing devices and the data requirements.
2. An open data app for end users of the transport or other London system – describe the application and the benefits it provides to the users, decompose the application into the main modules needed and describe the data used, produce a sketch of the interface (screen) the user would see on a web browser or mobile device.

3. Engineering role – select an engineer who uses computer programming as a significant part of their work. This may be from a case study or from dialogue with a working engineer. Describe their overall role and in more detail the type of problem they use computing/ programming for and the approach they use for solving problems.

Briefly talk through Resource C1: Reporting framework example – congestion charge (page 108). This provides students with a scaffold for either of the first two tasks above. Common systems that students should be familiar with and can be reported on if they do not have their own ideas are live bus arrival information or the Oyster Card system.

The source for the first task may be from their own experience of travel in London or from their Explore visit or other contact with engineers or case studies of engineers.

The source for the second task will be the Transport for London open data feeds or the London Data Store data sets or some other source of London open data. Both of these sources give suggestions for uses with some of the data sets which the students might take on.

The Engineering role task will be most appropriate if students have been able to meet an engineer and ask questions of them. If students have not already met an engineer, then you might arrange London Transport or STEM Ambassadors to visit for this lesson.

Differentiation

Lower ability students might be encouraged to produce an oral report based on diagrams or main headings rather than a longer written report. They might be supported with their choice of problem. They might also work in groups with higher ability students.

LESSON 1: COMPUTING LONDON SOLUTIONS

ACTIVITIES

MAIN 1

Identifying the problem.

Ask for quick feedback on this and ensure that the problems chosen are such that students are likely to be able to decompose them at least into their main elements and to identify the data needed.

Each group of students should be able to clearly state the problem, transport system, app or an engineer role, that they wish to report on. Particularly they should be able to state the data required by their system and what the source of the data is.

MAIN 2

Ask students to research and produce a team poster, presentation or report.

Students may produce whatever content they feel addresses the task but as a guide, for the Transport System or Open Data App the contents should be at least as described in the Reporting framework resource and for the Engineering role as in the Questions to ask an engineer resource.

Plenary

If time allows in the lesson students may present their results or this may be done in a later lesson or as a special event.

LESSON 1: COMPUTING LONDON SOLUTIONS

RESOURCE C1: REPORTING FRAMEWORK EXAMPLE – CONGESTION CHARGE



Your report on a transport system, engineering problem or your app design should contain at least the following sections:

1. The main role of the system
2. The problem being solved
3. The main computer applications within the system
4. Data needed by the system and how it is collected
5. Conclusion/Evaluation

The paragraphs opposite give an outline of the sort of information you might include in each section using the example of the congestion charge.

1. Describe the role of the system

Find the main TfL page for the system or a Wikipedia or other link to research from. The TfL page for the Congestion charge is at

tfl.gov.uk/modes/driving/congestion-charge

This will give you a short description of the system and links to many of the application examples.

For example, from the above link: "The Congestion Charge is an £11.50 daily charge for driving a vehicle within the charging zone between 07:00 and 18:00, Monday to Friday. The easiest way to pay the charge is by registering for Congestion Charge Auto Pay. There are a range of exemptions and discounts available to certain vehicles and individuals."

More information is on the Wikipedia page at **en.wikipedia.org/wiki/London_congestion_charge**

2. Describe the problem being solved

More web research. This may be better service to customers, need for more efficiency, cost savings etc. In the case of the congestion charge it was recognised how much delay and pollution were caused by traffic congestion in London so the charge was aimed at reducing congestion and persuading commuters to use public transport or other lower polluting forms of private transport such as cycling.

LESSON 1: COMPUTING LONDON SOLUTIONS

RESOURCE C1: REPORT FRAMEWORK EXAMPLE – CONGESTION CHARGE CONTINUED



3. Describe applications

The congestion charge page shows links for Paying the charge, Auto pay, Location check. Look into some of the applications on the website and get descriptions and screenshots. Produce a sketch of the screens on the web or mobile app for inputs and outputs for one or more of the apps.

4. Describe the data and how it is collected

Check that the data described will fulfil particular applications. Screenshots or sketches of screen designs showing data to be entered and the results will be of help with this. For the congestion charge: knowing if a car is in the congestion charge zone requires input of the car number plate through the automatic number plate recognition cameras. Knowing if a congestion charge has been input requires the driver to input the payment through the web or other method and then knowing which penalties are owed requires no further data, this comes from drivers who have not paid in time. At this basic level, students should be able to suggest the main data requirements. They are not required to identify every field in the real system.

5. Conclusion/Evaluation

How effectively is the system solving the identified problems? What else could be done, what could be improved? For the congestion charge, there are reports on the change in congestion over time and other aspects such as the move away from private to public transport. There are suggestions to expand the congestion charge zone.

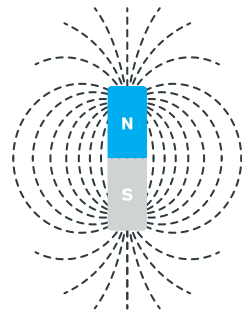
LINKS TO OTHER LONDON CURRICULUM SUBJECTS

The Connected City is part of London on the Move, a set of London Curriculum teaching resources that explore the application of STEM subjects in the transport systems of the city.



CHEMISTRY

London Refuelled takes a look at the chemistry that fuels London's transport and explores some of the future technologies that can help reduce pollution in the city.



PHYSICS

London's driving forces illustrates and explores the topics of energy and forces in the context of travel around the city and sets students the challenge of designing a vehicle for London's future.

CREDITS

The GLA would like to thank the following organisations for their contribution:

**Our collaborators on
the London Curriculum**

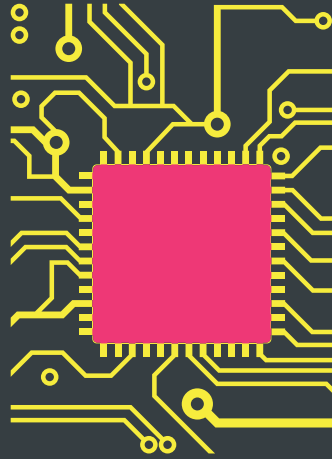


Copyright

Greater London Authority
May 2017

Greater London Authority
City Hall
The Queen's Walk
London SE1 2AA

www.london.gov.uk
enquiries 020 7983 4100
minicom 020 7983 4458



'The idea of using London as a teaching resource has never been explored much before, so both students and teachers are excited about it'

Key stage 3 teacher

'It makes me feel proud to be a Londoner'

Key stage 3 student